Multilevel modeling for data streams with dependent observations

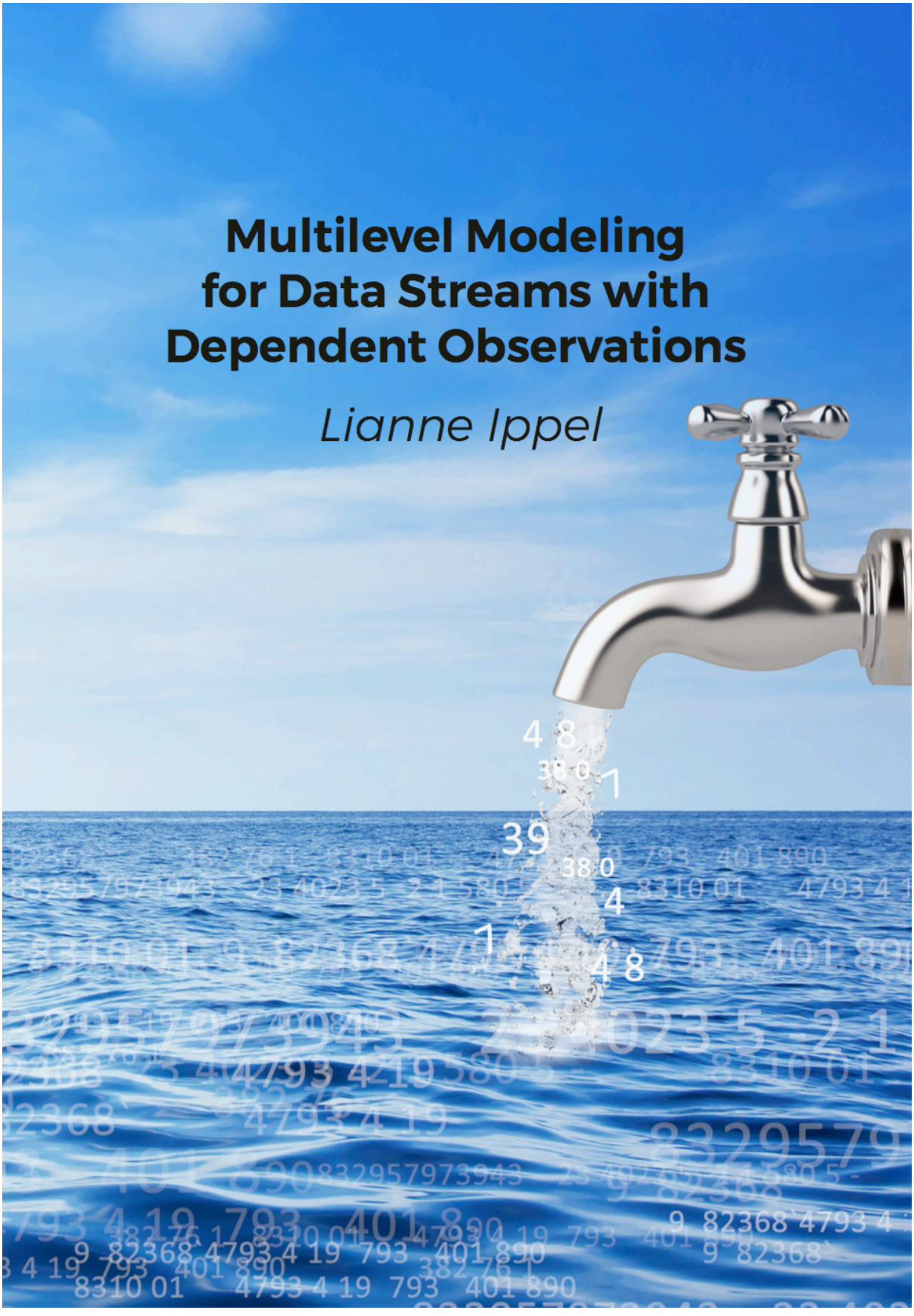
Ippel, L.

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Multilevel Modeling for Data Streams with Dependent Observations

Lianne Ippel



Multilevel Modeling for Data Streams with Dependent Observations

Lianne Ippel
Tilburg University

Printing was financially supported by Tilburg University.

ISBN:	978-94-6295-757-2
Printed by:	Proefschriftmaken | | Vianen
Cover design:	Faboosh design & art

Multilevel Modeling for Data Streams with Dependent Observations

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
Tilburg University op gezag van de rector magnificus,
prof.dr. E.H.L. Aarts,
in het openbaar te verdedigen ten overstaan van
een door het college voor promoties aangewezen commissie in
de aula van de Universiteit

op vrijdag 13 oktober 2017 om 10.00 uur

door

Gijsberdina Janna Elisabeth Ippel
geboren te Werkendam

Promotor:	prof.dr. J. K. Vermunt
Copromotor:	prof.dr. M.C. Kaptein
Overige leden van de Promotiecommissie:	prof.dr. G.J.P. van Breukelen
	prof.dr. M.E. Timmerman
	dr. M. Postma
	dr. M.A. Croon

Preface

One of my early-childhood memories comes from second grade at primary school. I am standing at the desk of my teacher, a five-year old and a bit too witty, asking my teacher when I would finally learn how to write and how to do math. Done with playing with blocks and dolls, I wanted to learn more! However, I had to wait one more year before I could start writing and calculating.

The eagerness to broaden my skills and deepen my knowledge has never left me. Years later, while finishing my Bachelor's degree in Sociology, I decided to develop myself even more and I applied for the research master at the faculty of Social and Behavioral Sciences.

I think it was not more than a month in the program, when Guy Moors approached me. He asked me which topic I wanted to study during my PhD project. Honored, and admittedly a little stressed out because I didn't feel like I had proven myself to be worthy of this position yet, we discussed several topics. Later in the program, I got the opportunity to work with Maurits Kaptein on my Master's Thesis. After the research master, he became my PhD supervisor in the following four years.

The book you are holding right now is the result of four years work. When I started this project, I never thought I was able to write the code, do the math, or have the writing skills to do this. Obviously, I have not accomplished the work on my own, but you will read more about that at the end of this book (Dankwoord).

Lianne Ippel
May, 2017

Contents

Preface **v**

1 Introduction **1**
1.1 The era of data streams . . . 1
1.2 Outline . . . 2
1.3 Contributions to the literature . . . 7

2 Dealing with Data Streams:
an Online, Row-by-Row, Estimation Tutorial. **9**
2.1 Introduction . . . 10
2.2 Dealing with Big Data: the options . . . 12
2.3 From Conventional Analysis to Online Analysis . . . 14
2.3.1 Sample mean . . . 14
2.3.2 Sample variance . . . 15
2.3.3 Sample covariance . . . 16
2.3.4 Linear regression . . . 17
Computation time of linear regression . . . 18
2.3.5 Effect size η^2 (ANOVA) . . . 18
2.4 Online Estimation using Stochastic Gradient Descent . . . 21
2.4.1 Offline Gradient Descent . . . 21
2.4.2 Online or Stochastic Gradient Descent . . . 23
2.4.3 Logistic regression: an Example of the Usage of SGD . . . 24
2.5 Online learning in practice: logistic regression in a data stream . . . 25
2.5.1 Switching to a safe well . . . 25
2.5.2 Results . . . 26
2.5.3 Learn rates . . . 26
2.5.4 Starting values . . . 27
2.6 Considerations analyzing Big Data and Data Streams . . . 28
2.7 Discussion . . . 30
Appendix 2.A Online Correlation . . . 31
Appendix 2.B Online linear regression . . . 32
Appendix 2.C Stochastic Gradient Decent – Logistic regression . . . 33
Appendix 2.D Wells data example . . . 34

3 Online Estimation of Individual-Level Effects using Streaming Shrinkage Factors **35**
3.1 Introduction . . . 36
3.2 Estimation of shrinkage factors . . . 39
3.2.1 The James Stein estimator . . . 39
3.2.2 Approximate Maximum likelihood estimator . . . 42
3.2.3 The Beta Binomial estimator . . . 43
3.2.4 The Heuristic estimator . . . 45
3.3 Predicting individual-level effects: when is the right time? . . . 45
3.4 Simulation Study . . . 47
3.4.1 Design . . . 47
3.4.2 Results . . . 48
3.5 LISS Panel Study: Predicting Attrition . . . 52
3.5.1 Results . . . 54
3.6 Conclusion and discussion . . . 56

4 Estimating Random-Intercept Models on Data Streams **59**
4.1 Introduction . . . 60
4.2 From offline to online data analysis . . . 62
4.3 Online estimation of random-intercept models . . . 63
4.3.1 The random-intercept model and its standard offline estimation 63
4.3.2 Online estimation of the random-intercept model . . . 65
4.4 Performance of SEMA evaluated by simulation . . . 68
4.4.1 Simulation study I: Evaluation of the precision of estimated parameters . . . 68
Design . . . 68
Results . . . 69
4.4.2 Simulation study II: Improving SEMA in low reliability cases . 72
Design . . . 72
Results . . . 75
4.5 An application of SEMA to longitudinal happiness ratings . . . 78
4.6 SEMA characteristics . . . 79
4.6.1 Theoretical considerations . . . 79
4.6.2 Convergence . . . 79
4.7 Extending SEMA . . . 80
4.8 Discussion . . . 82

5 Estimating Multilevel Models on Data Streams **85**
5.1 Introduction . . . 86
5.2 Offline estimation of multilevel models . . . 88
5.2.1 The offline E-step . . . 89
5.2.2 The offline M-step . . . 90
5.3 Online estimation of multilevel models . . . 91

5.3.1 The online E-step . . . 91
5.3.2 The online M-step . . . 94
5.4 Simulation study . . . 95
5.4.1 Design . . . 95
5.4.2 Results . . . 97
5.5 SEMA in action: predicting weight fluctuations. . . . 99
5.6 Discussion . . . 106

6 Discussion **109**
6.1 Overview . . . 109
6.2 Related approaches to analyze data streams . . . 109
6.2.1 Sliding window approach . . . 110
6.2.2 Parallelization . . . 111
6.2.3 Bayesian framework . . . 111
6.3 Data stream challenges . . . 112
6.3.1 Convergence . . . 112
6.3.2 Models used for analyses . . . 113
6.3.3 Missingness . . . 114
6.3.4 Attrition . . . 114
6.4 Null Hypothesis Significance Testing . . . 115
6.5 Future research directions for SEMA . . . 115

References **117**

Summary **127**

Samenvatting **129**

Dankwoord **131**

Chapter 1

Introduction

1.1 The era of data streams

In the last decade, technological developments have been rapidly changing our society. Instead of going out shopping in the city center we now often buy clothes in webshops, and instead of reading a newspaper once a day, we now continuously receive the headlines on our smartphones. While previously, it was often unknown who bought which products because it was difficult to trace individual customers, nowadays webpages can be designed to store all relevant digital transactions. As a result, these technological developments have led to an increase in digital information, which are collected on a large scale (Al-Jarrah, Yoo, Muhaidat, Karagiannidis, & Taha, 2015).

Analyzing the collected digital information might be challenging, because storing all the data requires a large computer memory. Additional to the memory burden, the fact that these observations keep *streaming* in complicates commonly used analyses even further, because the analyses often have to be redone when new observations enter to remain up to date. Situations where new data points are continuously entering and thereby augmenting the current data set are commonly referred to as *data streams* (Gaber, 2012).

When the data are arriving over time, it might be necessary to act upon the data while they enter: tailor the webpage to the currently browsing individual, warn patients to take their medication, or give people an extra nudge to respond to the questionnaire. Failing to act in real time might result in the potential customer leaving the webpage, because it did not appeal to him, the lack of medication could be deteriorating the patient's health, or a respondent failing to answer the questionnaire in time. These three examples clearly illustrate that in many situations failing to analyze the data in real time makes the analysis rather ineffective.

Digital data collection has also influenced the social sciences. Until recently, data were often collected by inviting respondents to fill out paper-and-pencil questionnaires. After a period of data collection, the resulting data set would be considered 'finished' and analyzed. Using modern technological innovations, data are nowadays commonly collected using web surveys or smartphone applications. Using

these digital approaches, it has become easier, cheaper, and faster to collect data from many individuals at the same time and to monitor these individuals over time.

Besides collecting more data using less resources, these developments have also created new opportunities to study individuals' behavior. Instead of asking for their typical behavior or feelings, which respondents would have to recall from memory, respondents are asked at random intervals to fill out some questions about their current feelings. This technique is called *Experience sampling* (see e.g., L. F. Barrett & Barrett, 2001; Trull & Ebner-Priemer, 2009) and commonly uses a smartphone application that gives a signal at random intervals to alert the respondent to answer the questionnaire. Experience sampling has become a common method to collect data in social science (Hamaker & Wichers, 2017) and, even though commonly not analyzed as such, the method does give rise to a data stream.

Analyzing data streams in real time is possible when fast prediction methods are available. Especially when data points stream in rapidly, the demand for more computational power to analyze the data in real time and the memory capacity to store all the data increases continuously. Even though computational power and memory capacity have grown substantially over the last decades, obtaining up-to-date predictions in a data stream is still a challenge. Due to the influx of data points, traditional methods which revisit all observations to update the predictions when new data have entered are bound to become too slow to be useful in a data stream.

In this thesis, approaches to analyze data streams in real time are studied and new methods are developed for the analysis of data streams consisting of dependent observations. These new methods facilitate the use of data stream applications encountered in the social sciences.

1.2 Outline

Figure 1.1 presents an overview of the structure of this thesis. Note that, Chapter 2 and Chapter 4 are published as separate journal articles and Chapter 3 and Chapter 5 are submitted for publication. This might have led to some repetition and inconsistencies in notation across the chapters. Below, a short illustration of the approach to analyze data streams is given, after which the topics (the 'branches' of Fig. 1.1) of each of the chapters (the 'leafs' of Fig. 1.1) are introduced.

A commonly used approach to analyze data streams is very intuitive. Let's imagine we are at a baseball field, and we want to keep scores of the teams. When a baseball player scores a point, we simply increment the score of the team who scored with one. This type of updating of the result of an analysis is referred to as *online learning* (Cappé, 2011a; Witten, Frank, & Hall, 2013). Using online learning, an analysis is done without returning to previous data points. Because online learning methods only store some *summary statistics* in memory, data points do not have to be stored in memory. The sum score is an example of a summary statistic: if we know the sum of the points scored, we can update this sum score by incrementing it with

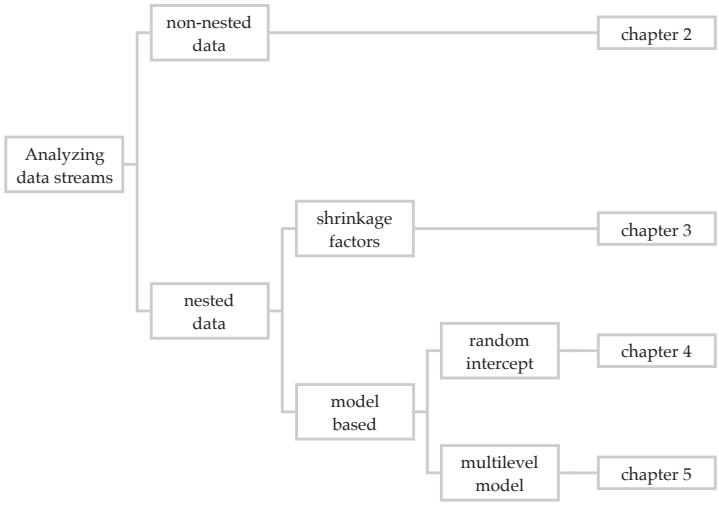


Figure 1.1: Graphical outline of this thesis

one when a baseball player scores a point. On the other hand, *offline* learning is an estimation procedure which uses all the observations in memory and revisits these observations when new data enter to update the result of an analysis. In an extreme case of the baseball match example, we would have to go back in time to rewatch the match again and count points over again, every time a new point is scored. While this example seems inefficient and perhaps rather odd, redoing analyses when new data arrive is currently common practice in many social science applications.

In **Chapter 2** (the first leaf of Fig. 1.1), a more detailed introduction to data streams and tools to analyze these data streams are discussed. The focus of this chapter is mainly on online learning. It is shown how simple parameters such as the sample mean but also more complex parameters such as the coefficients of a logistic model can be estimated in a data stream using online learning.

All the methods presented in this chapter share an important assumption, namely that the data points are independent. However, this assumption is likely to be violated in the context of data streams, due to the fact that the same individuals are observed repeatedly. Two observations of the same individual are likely to be more similar than two observations of two different individuals; hence, the data points are nested within individuals and are, as a result of that nesting, no longer independent. Violating this assumption results in more prediction error than when methods are chosen which do take the dependency into account. However, models that account for the dependency between the data points are much more complex to estimate. Thus far, most online learning methods do not take into account that the observations are nested. In this thesis, online learning methods which do account for the nesting in the data are developed and evaluated (branch 2: nested data, Fig. 1.1).

Let us return to the example of the baseball match and assume that we are now interested in who is the best baseball player. We could compute the average hitting proportion over all players easily online by counting the total number of hits by the total number of attempts; we call this an *aggregated* analysis. However, the aggregated analysis only gives us one estimate of the hitting proportion for all players, which does not answer our question who is the best player. So, it would be more appropriately to look at the individual batting behavior of the players. In order to answer our question, we could update the proportion of hits online for each player separately when they hit or miss the ball and the one with the highest proportion would be the best player. This approach, referred to as a *disaggregated* analysis, i.e., for each player separately, is straightforward to implement in a data stream. However, this disaggregated analysis is a naive approach to solve this problem. Stein (1956) showed that if there are more than two units, e.g., baseball players, just using a baseball player's hitting proportion does not result in the most accurate prediction of this players true batting ability. Instead, he proofed that the so-called *shrunken* estimates yield more accurate predictions than the observed individual averages. In terms of our baseball example: if we include the batting behavior of all players in predicting individual batting abilities, we are on average more accurate than using the observed individual hitting proportions.

The concept of shrinkage estimation is illustrated in Figure 1.2. The top of this figure presents the observed individual proportions and the bottom presents the shrunken estimates. The dotted lines connect the observed averages to the estimated abilities. The solid line is the overall average. As can be seen from Figure 1.2, the estimated abilities are shrunken closer to each other than the observed individual averages. It can be shown that these shrunken estimates predict more accurately the true ability than the individual average; i.e., on average is the difference between the predicted ability and the true ability smaller if you use a shrunken estimate instead of the observed average. Thus, if we want to predict player A's probability to hit the ball, then we should also take into account how well other players are doing. This rather counterintuitive finding of Stein (1956) is also known as Stein's paradox (Efron & Morris, 1977).

To illustrate Stein's paradox, let us assume that we are studying people's ability of throwing dice. We coin those who repeatedly have high score (sixes) "good" dice-throwers, while those that repeatedly have low scores are "poor" dice throwers. We, subsequently, invite 1,000 people to throw a dice twice, and we observe their scores. In our sample, we find 28 "good" dice-throwers; these people managed to throw a six twice in a row.

Now, Stein's paradox manifests itself when we use the historical data (hence, the two previous throws), to predict the future data. In our jargon above, the disaggregated analysis would lead us to predict a score of six, which most people immediately object to: the 28 'good dice throwers' were just lucky, and it is unlikely (or to be more accurate, the probability is $1/6^{th}$) that their next throw will be a six again. The

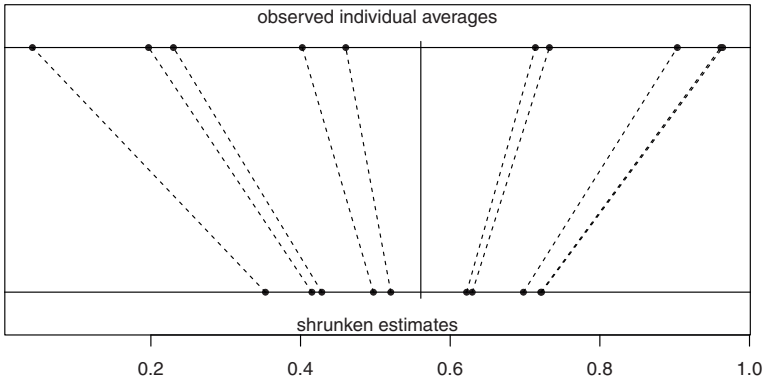


Figure 1.2: Graphical display of the effect of including other observed averages in estimating true abilities

aggregated analysis, on the other hand, leads us to predict an average score of about 3.5 (which was the average in our 1000 people sample) and seems more sensible in this case.

The fact that for dice-throwing it is seems intuitively feasible to look at the data of others to predict individual performance can be understood in terms of "signal" and "noise"; the signal, ones "dice-throwing-skill" is clearly non-existent, while the noise, the sheer "luck" of throwing two sixes in a row, is clearly driving the skill level of the 28 good throwers. Most people intuitively understand this noise should be corrected for in the case of dice throwing.

What is often underrated however, and provides an intuition to the origin of Stein's paradox, is that any measurement will contain both signal and noise to some extent. When there is clearly lot's of noise, we intuitively grasp that previous performance of an individual is not a good predictor, and that we rather want to use the scores of everyone else involved to get a better grasp of the underlying process. Oddly, when we move to baseball scores, many people seem to totally rule out such noise, and suddenly feel inclined to derive predictions solely based on the individual-level scores; Stein's shrinkage estimators provide a smooth weighting between the individual-level "skill" and the group scores, to correct for some of the noise introduced by the "best" batters merely being lucky.

How much the other players' averages influence the estimate of a single batting ability is determined by a *shrinkage factor*. W. James and Stein (1961) came up with one of the first shrinkage factors and since then multiple shrinkage factors have been developed, which differ in how much they shrink the observed individual averages towards the overall average and whether all observed individual averages are shrunken equally (Morris & Lysy, 2012). In **Chapter 3** four shrinkage factors are studied. For each of these four shrinkage factors an online approach is developed

such that the shrinkage factors are suitable to estimate the individual abilities during a data stream. The standard offline approach and the online approach are compared in a simulation study and are applied to an empirical example to predict which respondents would fail to respond to a questionnaire in a repeated-measurements design. While some shrinkage factors perform better than others, the accuracy of the predictions of the online and the offline estimated shrinkage factor are very similar.

Next, we turn to the last 'branch' of Figure 1.1: Analyzing data streams with nested data using a model-based approach. In social sciences, nested data are often analyzed using *multilevel models* (Demidenko, 2004; Raudenbush & Bryk, 2002), where we use the term *level 1* to refer to the observations and *level 2* to refer to the grouping variable. Using our baseball example, the batting observations are at level 1 and the baseball players are at level 2. Multilevel models have a number of advantages over traditional methods of analysis: e.g., unlike aggregated analyses, multilevel models take the nested structure of the data into account, and multilevel models consist of less parameters than the disaggregated analyses, which make the multilevel models easier to interpret.

Multilevel models are usually fitted to the data using an estimation framework called *Maximum Likelihood* (Myung, 2003). The aim of Maximum Likelihood estimation is to find the parameter values that maximize the likelihood of the observed data. However, unlike parameters such as the mean, the parameters of the multilevel model cannot easily be computed. In order to find those values for the parameters, one has to rely on some iterative procedure, such the *Expectation Maximization* algorithm (Dempster, Laird, & Rubin, 1977) or some Newton-type of algorithm (see, e.g., Demidenko, 2004). However, because these algorithms pass over the data repeatedly to find the Maximum Likelihood solution, the data points are stored in memory and revisited in each iteration. In addition, when used in a data stream, each time a new data point enters, the iterative fitting procedure has to be repeated again in order keep the parameters up to date. As a result, analyzing the data using this model in a data stream could become infeasible when data keep streaming in rapidly.

In **Chapter 4**, an alternative algorithm is developed, called *SEMA*, acronym for Streaming Expectation Maximization Approximation. In this chapter (see, Fig. 1.1), the focus is on the simplest multilevel model: the random intercept model (Raudenbush & Bryk, 2002). The SEMA algorithm fits a random intercept model while the data are entering, and more importantly, it does so without going back to the previous data points, which can then be discarded from memory. The SEMA algorithm is compared with the standard offline fitting procedure both in a simulation study and in an empirical study on respondents wellbeing. The SEMA algorithm is able to obtain parameter estimates, which are very similar to the estimates obtained by the offline procedure, both in the simulated data stream and in the empirical data stream, while SEMA is much faster.

The last 'leaf' of Figure 1.1 belongs to the same 'model-based' branch as the previous chapter. In **Chapter 5**, an extension of SEMA is presented. The random intercept

model is extended with both time-constant predictors (such as gender, which is unlikely to change over time) and time-varying predictors (such as a player's current self esteem, which is likely to vary over time). Additionally, the time-varying (level 1) predictors can have different effects depending on the individual (i.e., random slopes). In a simulation study, the SEMA algorithm is compared with the standard offline procedure, which shows that SEMA can analyze these simulated data streams well. In an empirical study about the fluctuations in individuals' weights, SEMA adequately predicts the weight of the individuals in the data stream.

1.3 Contributions to the literature

The contributions of this thesis to the literature are twofold: providing an introduction to data streams for social scientists, and developing new methods to analyze data streams. First, efficient approaches to implement commonly-used models by social scientists are illustrated. While intensive longitudinal data collection is becoming more popular in social sciences (Hamaker & Wichers, 2017), efficient approaches to analyze the data have to supplement these developments to make optimal use of the data stream. By introducing computationally-efficient methods to estimate well-known models, data streams become more accessible for social scientists.

Secondly, the state-of-the-art methods currently used to analyze the data streams often do not account for nested observations (e.g., Bottou, 2010; Cappé & Moulines, 2009; Neal & Hinton, 1998). While data streams are commonly used in computer sciences and marketing research, we focus on methods and models commonly used by social scientists. In this thesis, computationally-efficient approaches to multilevel modeling are developed to account for the nested structure commonly found in data streams. The content of this thesis covers, in part, both the literature of the social sciences and of the computer science. Admittedly, the focus is more on the former than on the latter: this thesis introduces data streams - which are an active area of study in computer science - to social scientists and, in addition, develops novel methods to account for nested observations in data streams, a problem common to the social sciences.

Chapter 2

Dealing with Data Streams: an Online, Row-by-Row, Estimation Tutorial.

Abstract

Novel technological advances allow distributed and automatic measurement of human behavior. While these technologies provide exciting new research opportunities, they also provide challenges: datasets collected using new technologies grow increasingly large, and in many applications the collected data are continuously augmented. These *data streams* make the standard computation of well-known estimators inefficient as the computation has to be repeated each time a new data point enters. In this chapter, we detail *online learning*, an analysis method that facilitates the efficient analysis of Big Data and continuous data streams. We illustrate how common analysis methods can be adapted for use with Big Data using an online, or "row-by-row", processing approach. We present several simple (and exact) examples of the online estimation and we discuss Stochastic Gradient Descent as a general (approximate) approach to estimate more complex models. We end this chapter with a discussion of the methodological challenges that remain.

This chapter is published as Ippel, L., Kaptein, M.C, & Vermunt, J.K. (2016) Dealing with Data Streams: an Online, Row-by-Row, Estimation Tutorial. *Methodology*, *12*(4), 124-138

2.1 Introduction

The ever-increasing availability of Internet access, smart phones, and social media has led to many novel opportunities for collecting behavioral and attitudinal data. These technological developments allow researchers to study human behavior at large scales and over long periods of time (Swendsen, Ben-Zeev, & Granholm, 2011; Whalen, Jamner, Henker, Delfino, & Lozano, 2002). Because more data are made available for research, these technological developments have the potential to advance our understanding of human behavior (L. F. Barrett & Barrett, 2001) and its dynamics. However, these novel data collection technologies also present us with new challenges: If (longitudinal) data are collected from large groups of subjects, then we may obtain extremely large datasets. These datasets might be so large that they cannot be analyzed using standard analysis methods and existing software packages. This is exactly one of the definitions used for the buzz-term "Big Data" (Demchenko, Grosso, De Laat, & Membrey, 2013; Sagiroglu & Sinanc, 2013): datasets that are so large that they cannot be handled using standard computing machinery or analysis methods.

Handling extremely large datasets represents a technical challenge in its own right, moreover, the challenge is amplified when large datasets are continuously augmented (i.e., new rows are added to the dataset as new data enter over time). A combination of these challenges is encountered when — for example — data are collected continuously using smart-phone applications (e.g., tracking fluctuations in happiness, Killingsworth & Gilbert, 2010) or when data are mined from website logs (e.g., research into improving e-commerce, Carmona et al., 2012). If datasets are continuously augmented and estimates are needed at each point in time, conventional analyses often have to be repeated every time a new data point enters. This process is highly inefficient and frequently forces scholars to arbitrarily stop data-collection and analyze a (smaller) static dataset. In order to resolve this inefficiency, existing methods need to be adapted and/or new methods are required to analyze streaming data. To be able to capitalize on the vast amounts of (streaming) data that have become available, we must develop efficient methods. Only if these methods are widely available we will be able to truly improve our understanding of human behavior.

Failing to use appropriate methods when analyzing Big Data or data streams could result in computer memory overflow or computations that take a lot of time. In favorable cases, the time to compute a statistic using standard methods increases linearly with the amount of data entering. For example, if computing the sum over n data points requires t time (where the time unit required for the computation is dependent on the type of machine used, the algorithm used, etc.), then computing the sum over $n+2$ data points requires $t+2c$ time, where c is t/n. Thus, the time increase is linear in n and is every increasing as the data stream grows. In less fortunate and more common cases, the increase in time complexity is not linear but quadratic, or

worse, amplifying the problems. Regardless of the exact scaling however, if the data are continuously augmented both the required computation time and memory use eventually will become infeasible

The aim of this chapter is to introduce *online learning* (or row-by-row estimation), as a way to deal with Big Data or data streams. Online learning methods analyze the data without storing all individual data points, for instance by computing a sample mean, or a sum of squares without revisiting older data. Therefore, online learning methods have a feasible time complexity (i.e., the time required to conduct the analysis) and they require a feasible amount of computer memory when analyzing data streams or Big Data. In the latter case, a very large static dataset is treated as if it were a data stream by iterating through the rows, without having all data points available in memory.

Online estimation methods continuously *update* their estimates when new data arrive, and *never revisit* older data points. Formally online learning can be denoted as follows:

$$\theta_n = f(\theta_{n-1}, x_n),$$

or equivalently and a shorthand

$$\theta := f(\theta, x_n), \tag{2.1}$$

which we will use throughout the chapter. In Eq. 2.1, θ is a set of sufficient statistics (not necessarily the actual parameters of interest), which is updated using a new data point, x_n. The second equation for updating θ does not include subscript n because we use the update operator ':=', which indicates that the updated θ is a function of the previous θ and the most recent data point, x_n.

A large number of well-known conventional estimation methods used for the analysis of regular (read "small") datasets can be adapted such that they can handle data streams, without losing their straightforwardness or interpretation. We provide a number of examples in this chapter. Furthermore, we will also introduce Stochastic Gradient Descent, a general method that can be used for the (approximate) estimation of complex models in data streams. For all the examples introduced in this chapter, we have made [R] code available at `http://github.com/L-Ippel/Methodology`.

This chapter is organized as follows: In Section 2.2, conceptual approaches for the estimation of parameters in Big Data and/or data streams are discussed, and we focus primarily on *online learning*, the method further illustrated in the remainder of this chapter. In Section 2.3, we illustrate how often-used estimators such as sample means, variances, and covariances, can be estimated using online learning. Here the benefits of online learning methods to deal with data streams are illustrated by comparing the computational times of online and offline estimation methods. We then, in Section 2.4, provide an introduction of Stochastic Gradient Descent (SGD) as a general (approximate) method to estimate more complex models in data streams.

Section 2.5 describes an example of an application of SGD in the social sciences. In Section 2.6 we detail some of the limitations of the online learning approach. Finally, in the last section, we discuss the directions for further research on data streams and Big Data.

2.2 Dealing with Big Data: the options

In the recent years, data streams and the resulting large datasets have received attention of many scholars. Diverse methods have been developed to deal with these vast amounts of data. Conceptually, four overarching approaches to handle Big Data can be identified:

1. sample from the data to reduce the size of the dataset,
2. use a sliding window approach,
3. parallelize the computation,
4. or resort to online learning.

The first option, to sample from the data, solves the problem of having to deal with a large volume of data simply by reducing its size. Effectively, when the dataset is too large to process at once, one could "randomly" split the data into two parts: a part which is used for the analyses and a part of the data that is discarded. Even in the case of data streams, a researcher can decide to randomly include new data points or let them "pass by" to reduce memory burden (Efraimidis & Spirakis, 2006). However, when a lot of data are available, it might be a waste not to use all the data we could potentially use.

Option two, using a sliding window, also solves the issue of needing increasingly more computation power by reducing the amount of data that is analyzed. In a sliding window approach the analysis is restricted to the most recent part of the data (Datar, Gionis, Indyk, & Motwani, 2002; Gaber, Zaslavsky, & Krishnaswamy, 2005). Thus, the data are again split into a part which is used for the analyses and a part which is not used for the analysis. The analysis part (i.e., also coined "the window") consists of the m most recent data points, while the second part contains older data which is discarded. One could see a sliding window as a special case of option 1, where the subsample only consist of new data points. When new data enter, the window shifts to include new data (i.e. a (partially) new subsample) and ignore the old data. Although a sliding window approach is feasible in computation time and amount of memory needed, the sliding window approach has the downside that it requires domain knowledge to determine a proper size of the window (e.g., determine m). For instance, when studying a rare event, the window should be much larger than in the case of a frequent event. It is up to the researcher's discretion to decide how large this window ought to be. Also, when analyzing trends, a sliding window approach might not be appropriate since historical data are ignored.

The third option, using parallel computing, is an often-used method to analyze static Big Data. Using parallel computing, the researcher splits the data in chunks, such that multiple independent machines each analyze a chunk of data, after which the results of the different chunks are combined (see, e.g., Atallah, Cole, & Goodrich, 1989; Chu et al., 2006; Turaga et al., 2010). This effectively solves the problem of memory burden by allocating the data to multiple memory units, and reduces the computation time of static datasets, since analyses which otherwise would have been done 'sequentially' are conducted 'parallel'. However, parallelization is not very effective when the dataset is continuously augmented: since all data are required for the analyses, computation power has to eventually grow without bound for as long as the dataset is augmented with new data. Also, the operation of combining the results obtained on different chunks of data might itself be a challenge.

In this chapter, we will focus on a fourth method: online learning (e.g., Bottou, 1999; Shalev-Shwartz, 2011). As introduced in the previous section, online learning uses all available information, but without storing or revisiting the individual data points. Online learning methods can be used in combination with parallel computation, (for instance, see Chu et al., 2006; Gaber et al., 2005), but here we discuss it as a unique method that has large potential for use in the social sciences. This method can be thought of as using a very extreme split of the data; the data is split into a part consisting of $n - 1$ data points, where n is the total number of observations, and only 1 data point on the other hand. Additionally, in online learning methods, the $n - 1$ data points are summarized into a limited set of sufficient statistics of the estimates of the parameters of interest, which take all relevant information of previous data points into account (Opper, 1998). The summaries required to estimate the parameters of interest (often the sufficient statistics) are stored in θ. Subsequently, θ is updated using some function of the previous θ and the new data point; historical data points are not revisited.

Note that in this chapter, we focus on the situation where parameters are updated using a single (most recent) data point. There are also situations where one rather uses a 'batch' of data points. This is known as *batch learning*. See for a discussion of batch learning in SGD, Wilson and Martinez (2003), or Thiesson, Meek, and Heckerman (2001) about choosing block (or batch) sizes for the EM algorithm.

The two characteristics of online learning – including all the data in the estimate and not revisiting the historical data – jointly make online learning a very suitable approach to analyze data streams. However two downfalls remain, like sliding windows, online learning also requires domain knowledge to judge which information should be gathered beforehand; the researcher needs to choose the elements of θ and their update functions up front. Second, although this issue is not unique for online learning, the researcher often needs to choose starting values for the elements of θ. In the next section, we further detail online learning and how to choose starting values by providing the online adaptation of a number of conventional statistics.

2.3 From Conventional Analysis to Online Analysis

In this section, we discuss online analysis by providing several examples of the online computation of standard (often computed *offline*) estimators. We discuss the online estimation of the following parameters:

1. the sample mean,
2. the sample variance,
3. the sample covariance,
4. linear regression models, and
5. the effect size η^2 (in an ANOVA framework).

The online formulations we discuss in this section are exact reformulations of their offline counterparts: the results of the analysis are the exact same whether one uses an offline or online estimation method. Note that for each of these examples, small working examples as well as ready-to-use functions are available on `http://github.com/L-Ippel/Methodology`.

2.3.1 Sample mean

The conventional estimation of a sample mean ($\bar{x}$) is computationally not very intensive since it only requires a single pass through the dataset,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.2}$$

However, even in this case, online computation can be beneficial. The online update of a sample mean is computed as follows:

$$\begin{aligned} \theta =& \{\bar{x}, n\}, \\ n_n =& n_{n-1} + 1 \\ \bar{x}_n =& \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n_n}, \end{aligned}$$

or equivalently,

$$\begin{aligned} \theta :=& \{\bar{x}, n\}, \\ n :=& n + 1, \\ \bar{x} :=& \bar{x} + \frac{1}{n}(x_n - \bar{x}). \end{aligned} \tag{2.3}$$

where we again use the update operator ':=' and start by stating the elements of θ that need to be updated: in this case these are n (a count) and $\bar{x}$ (the sample mean). Note that, appropriate starting value(s) for all the elements θ need to be chosen.

This also holds for all the other examples provided. In the case of the mean, one can straightforwardly choose $n = 0$ and $\bar{x} = 0$ as this starting point does not impact the final result – this, regretfully, will not generally hold. Also note that an online sample mean could also be computed by maintaining $n := n + 1$ and $Sx := Sx + x_n$, where Sx is the sum over x, as the elements of θ; in this case, the sample mean could be computed at runtime using $\bar{x} = \frac{Sx}{n}$. This latter method however a) does not actually store the sought for statistic as an element of θ, and b) lets Sx grow without bound, which might lead to numerical instabilities.

We implemented an example of the online formulation of the sample mean in [R] code, *mean_online()*, which can be found at `http://github.com/L-Ippel/Methodology/Streaming_functions`. This implementation is a ready-to-use update of the sample mean. Below we present [R] code, which gives a demonstration of the use of the online implementation of the sample mean. In the [R] language, the '#' denotes a comment.


```
> # create some data:
> # number of data points = 1000,
> # mean of the data is 5 and standard deviation is 2:
> N <- 1000
> x <- rnorm(n = N, mean = 5, sd = 2)
> # create an object for the results:
> res <- NULL
> # the res object is needed such that you can feed back
> # the updates into the function
> # are created within the function, at the first call
> for(i in 1:n)
+ {
+   res <- mean_online(input = x[i], theta = res)
+ }
>
```

2.3.2 Sample variance

In case of the sample variance (often denoted s^2) more is to be gained when moving from offline to online computation as the conventional method of computing a sample variance requires two passes through the dataset:

$$\begin{aligned} \hat{s}^2 =& \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \\ =& \frac{SS}{n-1}, \end{aligned} \tag{2.4}$$

where SS is the sum of squares. Here, the first pass is used to estimate the sample mean $\bar{x}$, while the second pass is used to compute the sum of squares.

A numerically feasible online method to compute a sample variance in a data stream is Welford's method (1962), which, to keep notation consistent, we denote as:

$$\begin{aligned} \theta =& \{\bar{x}, SS, n\}, \\ d =& x_n - \bar{x}, \\ n :=& n + 1, \\ \bar{x} :=& \bar{x} + \frac{1}{n}(x_n - \bar{x}), \\ SS :=& SS + d(x_n - \bar{x}). \end{aligned} \tag{2.5}$$

Note the use of auxiliary variable d which is used since the online update of the sum of squares uses both the deviation from the current sample mean as well as from the previous sample mean:

$$SS_n = SS_{n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n). \tag{2.6}$$

In order to obtain the actual sample variance, we compute $\hat{s}^2 = SS/(n-1)$. The function to compute the sum of squares is coined *SS_online* and *var_online* uses the online sum of squares function to compute the variance. In order to obtain the standard deviation directly, *sd_online* function can be used. Note that in order to compute the variance and the standard deviation, starting values of $n = 1$ and a $\bar{x} = x[1]$ (which is the first data point), are required due to the fact that the sum of squares are divided by $(n - 1)$. The values $\{n = 1, \bar{x} = x[1]\}$ are provided as default, in case the user does not provide starting values.

2.3.3 Sample covariance

Next we turn to the estimation of quantities which depend on multiple variables, for instance the sample covariance between x and y, which is often computed using:

$$\begin{aligned} \hat{s}_{xy} =& \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}), \\ =& \frac{SC}{n-1}, \end{aligned} \tag{2.7}$$

where SC is the sum of cross products. Again, making use of Welford's method (1962), we can estimate the sample covariance online:

$$\begin{aligned} \theta =& \{\bar{x}, \bar{y}, SC, n\}, \\ n :=& n + 1, \\ \bar{x} :=& \bar{x} + \frac{1}{n}(x_n - \bar{x}), \\ SC :=& SC + (y_n - \bar{y})(x_n - \bar{x}), \\ \bar{y} :=& \bar{y} + \frac{1}{n}(y_n - \bar{y}). \end{aligned} \tag{2.8}$$

Note that, contrary to the online computation of the sample variance, we do not need auxiliary variables in this case since we can alternate updating of $\bar{x}$ and $\bar{y}$. The choice of which of the two sample means is updated first, is arbitrary (Pebay, 2008). Similar to the case of the sample variance, to compute the sample covariance, we compute $\hat{s}_{xy} = SC/(n-1)$.

In Appendix 2.A we present [R] code to compute covariances and correlations online. Since computing a correlation entails the estimation of sample means, variances, and a covariance all of these are also included in this code snippet. For readers wanting to compute the sum of cross products, the covariance, or the correlation during their analysis we have implemented the online estimation procedures in [R], and these can be found in the Streaming_functions file on github as respectively *SSxy_online*, *cov_online*, and *cor_online*. Note that, unlike the previous statistics, these functions require 2 inputs, one for each variable.

2.3.4 Linear regression

In applied research, often the aim is to estimate group differences or the effect of a certain independent variable x on an dependent variable y. In such cases the computation of a sample mean of a sample variance will not necessarily suffice to answer the research question. One often-used approach to answer research questions about the relationship between one or more independent variables and one dependent variable, is fitting a linear regression model:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon, \tag{2.9}$$

where $\mathbf{y}$ is the vector containing the data of the dependent variable, β is a vector of the regression coefficients of the q independent variables (including an intercept), and $\mathbf{X}$ is the matrix $(n \times q)$ with observed data, including a column of 1's for the intercept. Finally ϵ denotes the error or noise. When assuming $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the regression coefficients β are conventionally estimated as follows:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}, \tag{2.10}$$

where $\mathbf{X}'$ denotes the transpose of $\mathbf{X}$.

Computing this row-by-row works as follows: We can define $\mathbf{A} = \mathbf{X}'\mathbf{X}$ and $\mathbf{B} = \mathbf{X}'\mathbf{y}$ and compute the update as follows:

$$\begin{aligned} \theta =& \{\mathbf{A}, \mathbf{B}\}, \\ \mathbf{A} :=& \mathbf{A} + \mathbf{x}_n\mathbf{x}_n', \\ \mathbf{B} :=& \mathbf{B} + \mathbf{x}_n y_n. \end{aligned} \tag{2.11}$$

To obtain the regression coefficients, β, one computes $\hat{\beta} = \mathbf{A}^{-1}\mathbf{B}$. This method is well known in the parallel computing literature and is, for example, described in

Chu et al. (2006).

Although fairly simple, computing the regression coefficients this way has a disadvantage: Every time that $\hat{\beta}$ is computed, a matrix inversion is required. Especially when the number of independent variables q is large, this itself can be a computationally intensive operation. We can address this by updating the inverse matrix, $\mathbf{A}_inv$, directly online, using the Sherman–Morrison formula (Escobar & Moser, 1993; Plackett, 1950; Sherman & Morrison, 1950):

$$\begin{aligned} \theta =& \{\mathbf{A}_{inv}, \mathbf{B}\}, \\ \mathbf{A}_{inv} :=& \mathbf{A}_{inv} - \frac{\mathbf{A}_{inv}\mathbf{x}_n\mathbf{x}_n'\mathbf{A}_{inv}}{1 + \mathbf{x}_n'\mathbf{A}_{inv}\mathbf{x}_n}, \\ \mathbf{B} :=& \mathbf{B} + \mathbf{x}_n y_n, \end{aligned} \tag{2.12}$$

where $\mathbf{A}_{inv}$ is the inverted matrix $\mathbf{A}$. In practice, one would use a small part of the data to create matrix $\mathbf{A}$, invert this matrix, after which the original matrix $\mathbf{A}$ can be discarded from computer memory. The "small" part of the data that is used should at least have $n > q+1$ for $\mathbf{A}$ to be non-singular. Obtaining the regression coefficients using Equation 2.12 only requires a matrix multiplication: $\hat{\beta} = \mathbf{A}_{inv}\mathbf{B}$. In Appendix 2.B, we implement online linear regression in [R] using Sherman-Morrison formula. At the github page mentioned before, the function is named *lm_online*. The function requires two separate inputs, one for the dependent variable, and one input for the independent variables. The latter can obviously be a vector of multiple variables.

Computation time of linear regression

To illustrate the difference between online and offline methods, Figure 2.1 presents a comparison of the computational time required to compute the estimates of the regression coefficients $\hat{\beta}$ in a data stream between the three estimation methods discussed above. The x-axis denotes the number of data points seen so far. While the scale of the y-axis (time) will heavily depend on the size of the model (the number of parameters q) and the type of computing system used, the qualitative results presented here will hold in general: the computational time needed to obtain an estimate of β, at each value of n, will grow quite quickly (quadratic) for the offline method, while it grows only slowly for the online methods (linear). This result clearly illustrates the computational benefits of online methods over offline methods. It can also be seen that the direct online computation of the inverted matrix $\mathbf{A}_{inv}$ is faster than inverting the matrix at each time-point. This latter difference however only affects the slope of the linear computation time.

2.3.5 Effect size η^2 (ANOVA)

In many studies in sociology and psychology, it is of interest to examine whether k distinct groups differ from one another, for instance because one group of participants received a treatment while the other group of participants did not. When such

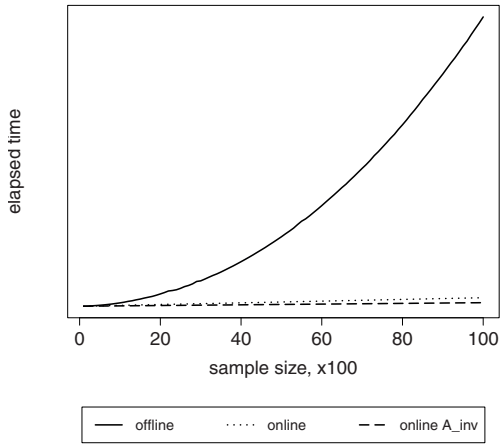


Figure 2.1: Computation time of regression coefficients using offline estimation (solid line), online estimation by inverting the matrix (online A_{inv}, dotted line), or online estimation by online updating the inverted matrix (dashed line).

experiments are carried out using modern interactive technologies, such as on social media platforms, sample sizes can grow very quickly. Traditionally, researchers often analyze the data from such group comparisons using an ANOVA approach. Between-subjects ANOVAs can be computed fully online. Here, we focus on the computation of the effect size η^2 which is given by:

$$\begin{aligned}\eta^2 =& \frac{SS_b}{SS_b + SS_w},\\ =& \frac{SS_b}{SS_t},\\ =& 1 - \frac{SS_w}{SS_t},\end{aligned} \tag{2.13}$$

where SS_b equals the sum of squares *between* the k groups, SS_w is the sum of the sums of squares *within* each of the k groups, and SS_t is the total sum of squares. The last expression of Equation 2.13 shows that computing both SS_w and SS_t in a stream suffices to compute the desired effect size.

Equation 2.6 already presented how sums of squares can be computed in data streams. The only complexity introduced in the ANOVA example is the computation of the sums of squares within each of the k groups. This requires computing the sample mean within group k:

$$\bar{x}_k := \bar{x}_k + \frac{x_{k,n} - \bar{x}_k}{n_k}. \tag{2.14}$$

which is only updated once a data point $x_{k,n}$ originates from group k. Subsequently, Equation 2.6 is used within each group k, substituting $\bar{x}$ with $\bar{x}_k$ to compute SS_w. The computation of the effect size or proportion of variance explained (η^2) in a data stream thus requires the following parameters:

$$\theta := \left\{ \begin{array}{r} \bar{x}_k, n_k \\ \bar{x}, n, SS_t, SS_w \end{array} \right\}, \tag{2.15}$$

where the top row of θ indicates the parameters at the *group* level (and hence need to be kept in memory for each group k) and the bottom row indicates the global parameters, which are only single parameters which need to be stored in memory. Thus, in total θ contains $2k + 4$ elements. We have implemented the online computation of η^2 in a function named *etasq_online*. This function will compute η^2 when two or more groups are available. Note that this function also requires two inputs: the data point and to which group this data point belongs. New groups can easily be included during the stream, without the data analyst interfering in the analysis.

In order to compute the F-statistic online, one can use the information that is already available:

$$F = \frac{(SS_t - SS_w)/(k-1)}{SS_w/(n-k)}. \tag{2.16}$$

It is important to note that repeated testing until a certain small p-value is found, which might be attractive if results are available for each new data point, is considered a questionable research practice (Armitage, McPherson, & Rowe, 1969; John, Loewenstein, & Prelec, 2012; Simmons, Nelson, & Simonsohn, 2011), due to inflation of Type 1 error. For instance, when a researcher decides to collect more data based on whether the ANOVA is significant 10 times with significance level of 5% while new data are entering, the actual Type 1 error equals

$$\alpha = 1 - (1 - 0.05)^{10} = 0.401,$$

instead of the 5% she started with. Perhaps the most common correction of this inflated Type I error is known as Bonferroni correction (Armstrong, 2014). Effectively this correction decreases 'α' as a function of the number of tests to prevent an increase of Type I error.

We will continue our discussion of online learning methods with Stochastic Gradient Descent (SGD). SGD is an optimization method which is useful to estimate more complex models when analytical solution are not available.

2.4 Online Estimation using Stochastic Gradient Descent

In the previous section, we have discussed how to estimate, fully online, a number of statistics and models that are often used in the analysis of sociological and psychological data. We have also illustrated the computational advantages of online estimation for very large datasets and data streams. However, for each of the methods discussed above we could derive exact summation methods; using simple algebra, it was possible to transform standard estimation methods to online variants. Unfortunately, this is not always the case. Many estimation methods, especially those that require multiple iterations through a static dataset, cannot exactly be implemented online, in part because even when using conventional offline analysis the estimation is approximate. Examples are logistic regression or multilevel models. However, this does not mean that we can only estimate very simple models online: there is a multitude of methods available for the online estimation of more complex models. In this section, we focus on Stochastic Gradient Descent (SGD), a general online estimation method that can be used for estimation of more complex statistical models.

To explain SGD, we will first discuss Gradient Descent (GD), an optimization method that is often used in conventional offline analysis and provides a logical starting point for SGD. We provide a general intuition to GD / SGD, and subsequently provide the technical details. Lastly, we will provide an applied example of fitting a logistic regression model using SGD, for which [R] code is provided in Appendix 2.C.

2.4.1 Offline Gradient Descent

There are multiple ways to obtain estimates for parameters of statistical models, for instance using a least squares approach (see, for example, Keith, 2014), using Maximum Likelihood estimation (ML), or using the method of moments (e.g., Arvas & Sevgi, 2012). In the social sciences we often use the maximum likelihood framework (see, for an introduction, Myung, 2003). In the maximum likelihood framework, we want to obtain the parameter values which maximize the probability of the data. Assuming independent observations, the likelihood function for many models takes the following form:

$$\mathcal{L}(\zeta|x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i|\zeta), \tag{2.17}$$

where ζ is a set of parameters, $f()$ is a probability density function (PDF) (or probability mass function in the discrete case), and as before $x_1, \ldots, x_n$ denote the observations. In words, Equation 2.17 states that the likelihood of ζ given the observed data is a product of the individual probabilities of each of the data points. In practice, it is often (much) simpler to obtain the maximum likelihood estimates by taking

the logarithm of the likelihood:

$$\ell(\zeta|x_1,\ldots,x_n) = \sum_{i=1}^{n} \ln f(x_i|\zeta), \tag{2.18}$$

which effectively replaces the product term with a sum, and gives the same solution for the maximum since the logarithm is a monotonic function. For some models, obtaining a maximum likelihood estimate analytically, after the log-likelihood, is defined is straightforward: we take the derivative of the log-likelihood, set it to zero, and solve for the parameters to obtain the required estimates. Effectively, this has already been demonstrated: the estimation of the sample mean, and of linear regression models as discussed in previous sections, actually *are* analytical maximum likelihood estimates given the appropriate models (see for example, Gelman & Hill, 2007).

However, exact analytical solutions are not always available. In such cases, one can resort to *approximate* methods, which are also frequently applied in offline analysis. One such approximate algorithm is called Gradient Descent, or actually Gradient *Ascent* because we use it in the context of a likelihood function which we want to *maximize*. Gradient Descent is the name most often used in the machine learning literature and classically used to *minimize* the error.

The GD algorithm can be stated as follows:

$$\zeta := \zeta + \lambda \nabla \ell(\zeta|x_1,\ldots,x_n), \tag{2.19}$$

where λ is a learn rate (also known as step size) chosen by the researcher and $\nabla l()$ denotes the *gradient* (vector of first order derivatives) of the log-likelihood function. Intuitively, this algorithm states that one chooses starting values for each parameter and evaluates the gradient using these values. In the simple case where ζ is scalar, and the gradient simplifies to the derivative, this evaluation gives information regarding the slope of the log-likelihood function: if the slope is positive[2] then the maximum can be found at higher values of ζ and we can make a step towards higher values of ζ. If the slope at ζ is negative, we need to step in the opposite direction: we need to choose a lower value. Using this intuition, GD iteratively – passing through the dataset multiple times – takes steps towards the maximum of the log-likelihood function. In the case that ζ is a vector, GD takes a step in q dimensions: for each parameter (i.e., dimension) GD determines whether the slope of the derivative is positive or negative, accordingly GD takes a step in the q dimensional space which causes the steepest ascent towards the maximum of the likelihood function.

Figure 2.2 provides an illustration of a gradient in a single dimension (e.g., the derivative). On the x-axis are possible parameter values and on the y-axis is the likelihood. The dashed curve is the likelihood of a given parameter value. At each point on the curve we can evaluate the first order derivative. Figure 2.2 presents

[2]Here, we are assuming the log-likelihood function to be well-behaved.

three evaluations of the first order derivative, including the tangent lines at each of the three points (solid black lines). When the derivative has a positive number, the likelihood increases by increasing the parameter value. Opposite, if the derivative has a negative value, the slope is negative, and the likelihood increases by decreasing the parameter value. Obviously, the aim is to find the parameter value where the derivative is equal to zero, in order to find the maximum. The second evaluation of the derivative in Fig. 2.2 contains two dotted lines. These dotted lines illustrate how the next evaluation is chosen. GD increases the value of the parameter when the result of the evaluation is positive and visa versa when the result is negative. The magnitude of the derivative together with the learn rate influence how much the parameter value is changed.

Gradient Descent can be a very effective method of finding the maximum likelihood value of ζ, although it is not without difficulties. For example, the parameter λ controls the size of the steps and has to be chosen carefully: a learn rate which is too large can be problematic since the algorithm could make jumps over the maximum likelihood solution. A learn rate which is too small causes the algorithm to take very small steps, and thus many iterations will be needed to obtain the maximum likelihood estimate. It depends on the model (e.g., complexity of the model, complexity of the likelihood function, etc.) what learn rate will be appropriate. One can choose for either a *fixed* learn rate or a learn rate which is adaptive, for instance one could choose to let the learn rate decrease with the number of iterations. A more extensive discussion on choosing the appropriate learn rate for complex models can be found in Wilson and Martinez (2003) and Bottou (2010).

2.4.2 Online or Stochastic Gradient Descent

Gradient Descent provides an iterative, approximate method to find maximum likelihood estimates. Deriving an effective *online* version of GD is as follows: instead of iterating over the full dataset multiple times and updating ζ each iteration, the algorithm takes a small step to a more likely parameter value every time a data point enters:

$$\zeta := \zeta + \lambda \nabla \ell_n(\zeta | x_n), \tag{2.20}$$

where we use ℓ_n to denote that we are evaluating the log-likelihood function. Hence, instead of updating the estimates of the parameters based on iterations using all data, we update based on each arriving data point. SGD will converge to an unbiased estimate of the parameters as long as the order in which the data points arrive is random (Bottou, 2010). This means that the process that generates the data, does not change over the period in which the data are arriving.

Note that in the case that the dataset is no longer augmented, SGD can still be a useful tool: Analyzing static Big Data using SGD circumvents that the entire dataset needs to be available in memory. By simulating that the data enter a point at a time

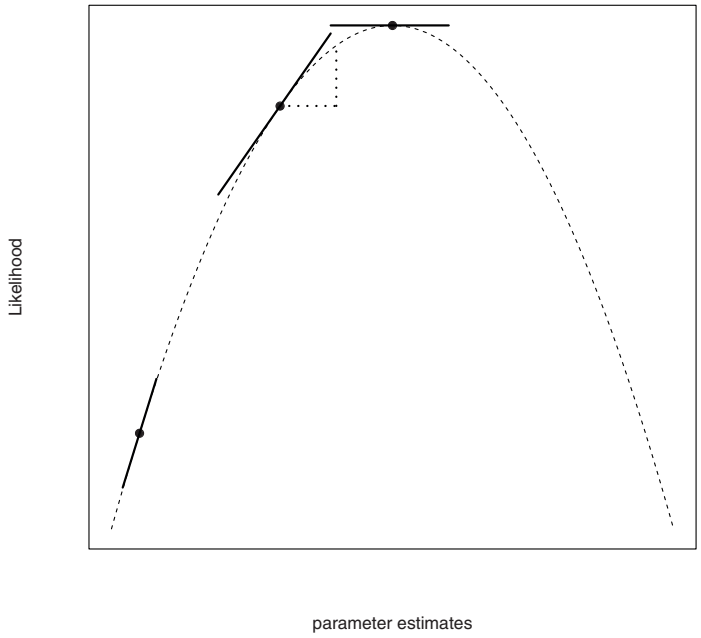


Figure 2.2: Graphical display of the likelihood function. In cases where the direct maximization of the likelihood function is difficult, an algorithm such as GD can be used to find the maximum. GD uses the slope of the tangent and a learn rate to make steps towards the maximum of the function.

and letting the data stream in repeatedly, SGD can obtain unbiased estimates while still estimating the parameters without seeing all data at once.

2.4.3 Logistic regression: an Example of the Usage of SGD

We present an example to illustrate SGD in which we are interested in the effect of independent variables on a binary dependent variable. In applied research, dependent variables are often binary, examples include whether and how people intent to vote (democratic versus republican, Anderson, 2000), or whether or not people smoke cigarettes (Emmons, Wechsler, Dowdall, & Abraham, 1998). In the case of a binary dependent variable, often a logistic regression model is chosen to describe the relationship between a binary dependent variable and continuous independent variables:

$$Pr(y = 1|\mathbf{X}) = p(\mathbf{X}) = \frac{\exp(\mathbf{X}\beta)}{1 + \exp(\mathbf{X}\beta)}, \tag{2.21}$$

where p is the probability to score a 1 on y, $\Pr(y = 1|\mathbf{X})$ and is modeled as a function of $\mathbf{X}$. Unlike linear regression, logistic regression does not have a closed-form solution to estimate the parameters β using a maximum likelihood approach, and hence even for offline analysis approximate methods are used. Estimating the parameters online can be done using SGD as follows. First, we specify the log likelihood:

$$\ell(\beta) = \sum_{i=1}^{n} y_i \log p(\mathbf{x}_i) + (1 - y_i) \log (1 - p(\mathbf{x}_i)). \tag{2.22}$$

Second, we compute the gradient (see, for more details for instance Agresti, 2002):

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} (y_i - p(\mathbf{x}_i))\mathbf{x}_i, \tag{2.23}$$

which in the case of offline estimation would be evaluated for all the data at once. When we use SGD to estimate the β's, the following online algorithm is obtained:

$$\begin{aligned} \theta =& \{\hat{\beta}, \lambda\}, \\ \lambda :=& \lambda + f(\lambda, x_n), \\ \hat{\beta} :=& \hat{\beta} + \lambda(y_n - p(\mathbf{x}_n))\mathbf{x}_n. \end{aligned} \tag{2.24}$$

Here we include λ, the learn rate, in θ. The inclusion of λ will not be not be necessary for a fixed value of λ, but it highlights that the learn rate could be a function of the data stream. Given an appropriate choice of λ, and a large enough data stream, SGD will correctly estimate the parameters of interest (Bottou, 2010). See Appendix 2.C for an implementation of SGD for the estimation of logistic regression in [R]. We implemented SGD for logistic regression in [R], the function is called *sgd_log* and can be used to estimate logistic regression in a stream.

2.5 Online learning in practice: logistic regression in a data stream

2.5.1 Switching to a safe well

To illustrate a logistic regression in a data stream, we use an example dataset, described in Gelman and Hill (2007). The dataset contains information regarding households in Bangladesh and whether or not they switch to a safe well to collect water. The wells were labeled safe if the arsenic level was low enough. Five years after the labeling of the wells, researchers collected data to study how many households had switched from their own unsafe well to another safe well. Switching to another well was dependent on whether owners of a safe well were willing to share their safe well and whether the households that did have an unsafe well were willing to make some extra effort to go to the safe well. The relatively small dataset consists of $N = 3020$ households. Among other variables, the dataset includes the distance in

meters to a safe well (X_{dist}) from the household, and arsenic level that is present in the water (X_{ars}).

In practice, choosing which variables to include from often a large set of variables, could be a challenging task on its own. Methods to deal with variable selection in a data stream are for instance the online Lasso (Yang, Xu, King, & Lyu, 2010) or based on Ridge regression (Tarrès & Yao, 2014). For the current example, we simulate that the data enter a point at a time by analyzing the data row-by-row using both offline and online implementations to predict whether the household switched to a safe well (coded 1) or did not switch (coded 0). The model we estimate contains two independent variables and an interaction term:

$$Pr(y = 1|X_{dist}, X_{ars}) = \frac{\exp(b_0 + b_1 X_{dist} + b_2 X_{ars} + b_3 X_{dist} X_{ars})}{1 + \exp(b_0 + b_1 X_{dist} + b_2 X_{ars} + b_3 X_{dist} X_{ars})}, \quad (2.25)$$

We thus estimate the four coefficients b_0, b_1, b_2, and b_3. The starting values for all four β's are zero, see Appendix 2.D.

2.5.2 Results

In Figure 2.3, we present the results of fitting a logistic regression in a data stream with four coefficients and an adaptive learn rate, $\lambda = \frac{1}{\sqrt{n}}$. The x-axes present the data stream and the y-axes present the estimated parameter values. During the stream we monitored the estimated parameter values using a moving average of 100 estimates. These moving averages are presented in Figure 2.3.

The estimates of the effect of the arsenic level (b_2) and the interaction term (b_3) are very accurate from the beginning of the data stream. The estimates of the intercept (b_0) and the effect of the distance to the next safe well (b_1) require some more data. The dashed line is fluctuating, even towards the end of the dataset: this is due to the fact that the learn rate is still quite large ($\frac{1}{\sqrt{3020}} = 0.018$) for a dataset this size. A smaller learn rate (or one that decreases more rapidly) would stabilize the SGD algorithm more, but increases the risk of introducing more bias.

2.5.3 Learn rates

To gain some insight in the sensitivity of SGD to its learn rates we also present the results of 4 different rates: .1, .01, .001, and $1/n$. We present the results of the four learn rates for the intercept, though the learn rates were equal for all coefficients. Again, the x-axis presents the data stream and the y-axis the estimated parameter value. Figure 2.4 presents the moving average of 100 estimates during the stream. Clearly, the curve with the largest learn rate shows much more fluctuation. Much of this fluctuation is already gone when we lower the learn rate to .01, although the online estimation of the intercept remains close to the offline estimation of the intercept. All fluctuation has gone for the two smallest learn rates. These two are a

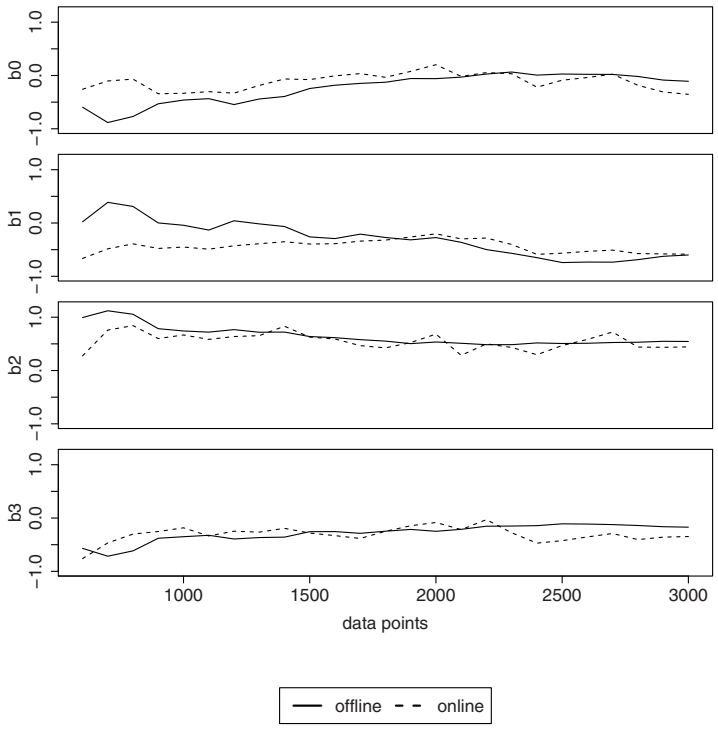


Figure 2.3: Online (dotted) and offline (solid) estimated beta coefficients of logistic regression as more data enter.

clear example of learn rates that are too low. In such cases the estimates do not, or hardly change.

2.5.4 Starting values

Lastly, we present the results of start the analysis with different starting values in Figure 2.5. On the x-axis is the data stream presented, y-axis is the estimated parameter value, and the lines are the moving average of 100 estimates. While the intercept had starting values {-2, -1, 1, 2}, the remaining coefficients had starting values equal to zero and the learn rate remained $1/\sqrt{n}$. Here we present the influence of the starting values on the final parameter estimate. Although there is some difference between the four lines, all four of them result in very similar parameter estimates. This illustrates that SGD does not really depend (given an appropriate learn rate) on the starting values and that the data dominate the results quickly.

For larger datasets and for continuous streams, which is what we primarily focused on in this chapter, the performance of SGD is often accurate.

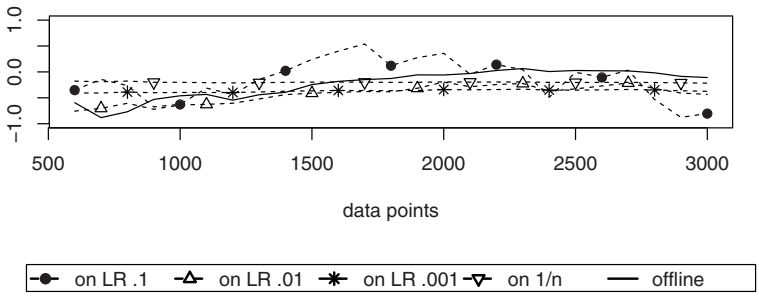


Figure 2.4: Online (dotted) and offline (solid) estimated intercepts for learn rates: .1, .01, .001, and $1/n$ coefficients of logistic regression as more data enter.

2.6 Considerations analyzing Big Data and Data Streams

In this chapter, we have discussed online learning as a way to deal with Big Data. However, some important issues remain. Here, we discuss two practical and two conceptual issues related to analyzing Big Data.

Practically, it has to be noted that at this moment not many off-the-shelf statistical packages are available to actually analyze data streams. The currently available software, for instance (and not exhaustive) *Apache Storm* (Toshniwal et al., 2014), *Apache Spark* (Karau, Konwinski, Wendell, & Zaharia, 2015), *RStorm* (Kaptein, 2014), *S4* (Neumeyer, Robbins, Nair, & Kesari, 2010), *RapidMiner* (M. Hofmann & Klinkenberg, 2013), *KNIME* (Berthold et al., 2009), and *MOA*, (Bifet, Holmes, Kirkby, & Pfahringer, 2010) often require extensive programming knowledge and focus mainly on the infrastructure of analyzing large datasets. There is still a large gap between the methods and software developed by computer scientists, and those that can be used by social scientist to analyze their data streams using models that they are accustomed to.

Second, we have to stress that for the application of online methods the analyst has to know beforehand what type of analysis and model is required to answer the research question. Online learning methods make use of a limited set of quantities – referred to the elements of θ throughout this text – to store the relevant information and to subsequently estimate model parameters. This means that it is important to know what information is required *before* the analysis. Any information that is not stored is forgotten and is impossible to retrieve if the data themselves are not stored.

A solution to this latter issue could be to run simultaneously different analyses and/or models, such that at a later point in time a decision can be made which analysis or model to use. This, of course, does require that enough computer memory is available to store the sufficient statistics of multiple models. A frequently adopted

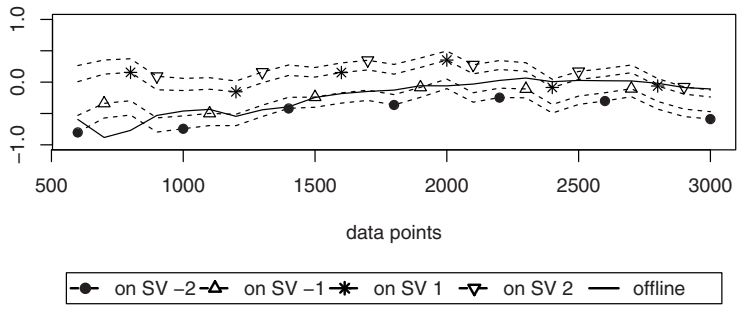


Figure 2.5: Online (dotted) and offline (solid) estimated intercepts for starting values: -2, -1, 1, and 2 coefficients of logistic regression as more data enter.

practical solution to this in the computer science literature is to adopt a so-called λ-architecture (Marz & Warren, 2013): the data stream is operated on online (for those computations that where specified in advance), but also stored and can thus be analyzed offline at a later point in time (often using parallelization methods to deal with the size of the dataset).

From a conceptual point of view, we do explicitly mention that we are not promoting repeated null hypothesis significance testing in data streams; this should be avoided. When a researcher decides to stop the data collection once she obtains a significant result of the hypothesis test, the Type I error rate increases above nominal level (i.e., too many false positives, Strube, 2006). It is considered a questionable research practice to repeatedly test for a significant effect and stop data gathering once the effect yields a $p < .05$ (John et al., 2012). When adopting an online learning approach, we encourage researchers to focus on obtaining precise estimates of the size of the effects of interest, in adherence to the APA guidelines (Wilkinson & Task Force on Statistical Inference, 1999), as opposed to null hypothesis testing.

Finally, it is not always feasible to translate all analyses from the offline framework to the online framework. For instance, the analysis of binary dependent data, data that are nested within units, which are in the offline case often analyzed using logistic multilevel (or random effects-) models, have not yet found a proper online synonym. Therefore, future research should be aimed at translating complex models, such as logistic multilevel models, to the online learning framework. Note that active research work is carried out in this field, with for example recent publications describing online approximations of the well-known Expectation-Maximization algorithm (Cappé & Moulines, 2009).

2.7 Discussion

Using new data collection methods and technologies, for instance experience sampling (L. F. Barrett & Barrett, 2001), to collect social and psychological data have made data streams more apparent and more prevalent in recent years. In this chapter, we discussed how social scientists can deal with these large datasets, and how regular estimation methods can be applied in the context of continuous data streams. We hope to have contributed to opening up the possibilities to answer both existing research questions as well as new types of research questions using large datasets or continuous data streams. Note that we have only touched upon a few methods which are used to analyze data streams; there are many more techniques available to analyze data streams, for instance the Bayesian framework can in some cases also be used to update the estimated parameters (e.g., Gelman, Carlin, Stern, & Rubin, 2004). In the case of conjugate priors, the posterior can be updated relatively easily. However, in the situation where the prior is not conjugate, other methods such as particle filtering are required to update the posterior (Robert, 2015).

Despite the versatility of online methods as displayed in this tutorial, many challenges remain: common methods such as (latent) factor analysis, mixture models, or multilevel models are not easily estimated online (See for a discussion and online approaches for multilevel models, Ippel, Kaptein, & Vermunt, 2016b). A possible way to deal with these types of analyses is to alter for instance the EM algorithm (Dempster et al., 1977). Suggestions for parallel computations and more efficient procedures for the EM algorithm have already been proposed (Cappé & Moulines, 2009; Neal & Hinton, 1998; Wolfe, Haghighi, & Klein, 2008), and this work should be extended to make the EM algorithm applicable for streaming data.

We hope that this chapter motivates applied researchers to explore new research areas that are opened up by the technological opportunity to monitor individuals in a data stream. We believe that data streams can provide social scientists with many new insights in human behavior and can provide new research areas to study human emotions and attitudes.

2.A Online Correlation

```
> N <- 1000                              #number of observations
> x <- rnorm(N, 5,2)                     #generate data
> y <- 1.5*x+rnorm(N)
> # because a correlation requires
> # at least 2 points we start with n=1
> n = 1; xbar = x[1]; ybar = y[1]; SC = 0; SSx = 0; SSy = 0;
> for (i in 2:N)
+ {
+    dx   <- (x[i]-xbar)                 #deviance x
+    dy   <- (y[i]-ybar)                 #deviance y
+    n    <- n+1                         #update number of observations
+    xbar <- xbar+(x[i]-xbar)/n          #update mean x
+    SSx  <- SSx+dx*(x[i]-xbar)          #update sum of squares for x
+    SC   <- SC+(x[i]-xbar)*(y[i]-ybar)#update sum of cross products
+    Sxy  <- SC/(n-1)                    #compute covariance
+    ybar <- ybar+(y[i]-ybar)/n          #update mean y
+    SSy  <- SSy+dy*(y[i]-ybar)          #update sum of squares for y
+    sx   <- sqrt(SSx/(n-1))             #estimate std.dev. x
+    sy   <- sqrt(SSy/(n-1))             #estimate std.dev. y
+    rxy  <- Sxy/(sx*sy)                 #estimate correlation
+ }
>
```

2.B Online linear regression

```
> N  <- 1000    # generate data
> x0 <- rep(1, N)
> x1 <- rnorm(N, 5,2)
> x  <- matrix(c(x0,x1),nrow=N)
> y  <- 3+1.5*x[,2]+rnorm(N)
> A  <- matrix(0,nrow=2,ncol=2); B <- c(0,0)
> #the as.matrix and as.numeric are required to get [r] running
> for (i in 1:N)
+ {             #fit linear regression:
+    if(i<3)
+    {          #update A as long as it is not invertible
+      A     <- A+x[i,]%*%t(x[i,])
+    }          #update B
+    B       <- B + as.matrix(x[i,])%*%y[i]
+    if(i==3)
+    {          #invert A when n>p
+      A_inv <- solve(A)
+    }
+    if(i>=3)   #update inverted matrix A_inv
+    {          #C is a scalar
+      C     <- as.numeric((1+x[i,]%*%A_inv%*%x[i,]))
+      A_inv <- A_inv - ((A_inv%*%x[i,]%*%x[i,]%*%A_inv)/C)
+      beta  <- A_inv%*%B #compute coefficients
+    }
+ }
>
```

2.C Stochastic Gradient Decent – Logistic regression

```
> N    <-3000
> x    <-rnorm(N,1,1)
> e    <-rnorm(N)
> y    <-rbinom(N,1, (exp(-2+1.5*x+e)/(1+exp(-2+1.5*x+e))))
> beta <- c(0,0)
> for(i in 1:N)
+ {
+    p    <- exp(beta[1]+beta[2]*x[i])/(1+exp(beta[1]+beta[2]*x[i]))
+    beta <-beta + lambda*(y[i]- p) %*%c(1,x[i])
+ }
>
```

2.D Wells data example

```
> beta    <- c(0,0,0,0)
> for(i in 1:nrow(wells.dat))
+ {
+    n    <- n+1
+    x    <- c(1,wells.dat[i,c(dist, ars, dist*ars)])
+    y    <- wells.dat$switch[i]
+    p    <- exp(sum(beta*x))/(1+exp(sum(beta*x)))
+    beta <-beta + 1/sqrt{n}*(y - p) %*%x
+ }
>
```

Chapter 3

Online Estimation of Individual-Level Effects using Streaming Shrinkage Factors

Abstract

In the last few years, it has become increasingly easy to collect data from individuals over long periods of time. Examples include smart-phone applications used to track movements with GPS, web-log data tracking individuals' browsing behavior, and longitudinal (cohort) studies where many individuals are monitored over an extensive period of time. All these datasets cover a large number of individuals and collect data on the same individuals repeatedly, causing a nested structure in the data. Moreover, the data collection is never 'finished' as new data keep streaming in. It is well known that predictions that use the data of the individual whose individual-level effect is predicted in combination with the data of all the other individuals, are better in terms of squared error than those that just use the individual mean. However, when data are both nested and streaming, and the outcome variable is binary, computing these individual-level predictions can be computationally challenging. In this chapter, we develop and evaluate four computationally-efficient estimation methods which do not revise "old" data but do account for the nested data structure. The methods that we develop are based on four existing shrinkage factors. A shrinkage factor is used to predict an individual-level effect (i.e., the probability to score a 1), by weighing the individual mean and the mean over all data points. In a simulation study, we compare the performance of existing and newly developed shrinkage factors. We find that the existing methods differ in their prediction accuracy, but the differences in accuracy between our novel shrinkage factors and the existing methods are small. Our novel methods are however computationally feasible in the context of streaming data.

This chapter is submitted as Ippel, L., Kaptein, M.C, & Vermunt, J.K. Online Estimation of Individual-Level Effects using Streaming Shrinkage Factors

3.1 Introduction

Researchers often encounter *grouped* data where the outcome variable of interest is *binary*. For example, Murnaghan, Sihvonen, Leatherdale, and Kekki (2007) compared the smoking behavior (smoking versus none smoking) of students that are grouped within different schools. Quintelier (2010) studied the effect of schools on students voting behavior (vote versus no vote), and Linares et al. (2010) monitored children over a long period of time (repeated measurements nested within children) to investigate the effect of air pollution on the presence (or absence) respiratory symptoms. Furthermore, Cheng and Cantú-Paz (2010) studied 'click' behavior in e-commerce, i.e., whether an individual clicks on an advertisement on a website. In this latter case, the repeatedly observed click-through behavior is nested within the individual. In each and every instance above researchers are interested in obtaining good estimates of the probability of an event occurring at the level of the individual, while respecting the nested structure in the data. In this chapter, we examine efficient methods of obtaining such estimates in a situation where the collected data arrive continuously and datasets are rapidly augmented.

To settle for an unambiguous terminology throughout, we adopt the terms of the latter e-commerce example. Here, a researcher could be interested in the individual-level effect μ_i, which is the estimated probability that an individual will click. Note that we use i to index the grouping factor which, in this particular case of multiple observations nested within the individual, denotes the individual customer whose click-through rate is being estimated. However, the methods discussed in this chapter do not restrict themselves to the nesting of observations that are nested within individuals but could also be used for groupings such as individuals within schools or schools within districts. Our interest lies in estimating the individual-level effect μ_i, accurately and computationally efficiently.

In a now classical paper, Stein (1956) showed that predicting the individual-level effects of one individual (i.e., μ_i) using only the data of this particular individual, thus without taking the other individuals into account, results in a larger average squared prediction error than when these other individuals are taken into account. He demonstrated that combining the estimated mean of an individual, which we denote p_i, with the estimated sample mean over all data points, $\bar{p}$, results in better out-of-sample predictions (see, for instance, Efron & Morris, 1977). Following this result, W. James and Stein in 1961 introduced the idea of a *shrinkage factor*, a way to weigh the estimated mean of an individual and the mean over data points to obtain a prediction of μ_i. The resulting weighted combination can be denoted as follows:

$$\hat{\mu}_i = (1 - \hat{\beta})p_i + \hat{\beta}\bar{p}, \tag{3.1}$$

where β is the so-called *shrinkage factor*. Because we focus on binary outcomes, the p_i in our case denotes the proportion of (for instance) clicks. In the remainder of this chapter we refer to p_i as the *individual mean*, and $\bar{p}$ as the *group mean*.

The aim of this chapter is to develop and evaluate different shrinkage factors which can be used to efficiently estimate the individual-level effect in a situation where new data present themselves over time. We refer to this situation as a *data stream*. In a data stream, the data collection is never "finished", for instance in click-behavior data on a website. In the case of real-time prediction, where up-to-date predictions of the individual-level effects are required at each moment during the stream, methods that can *update* rather than *re-estimate* the individual-level effects, greatly improve the speed of the estimation process (Pébay, Terriberry, Kolla, & Bennett, 2016).

In general, various methods are available to deal with data streams. For instance one could subsample from the data stream (i.e., at random include some of the data points in the analysis while excluding others), and analyze the subsample in order to obtain predictions (Efraimidis & Spirakis, 2006). While this method solves the problem of a growing dataset, it inherently limits the information and risks not being able to include data of specific individuals who are of future interest. Another method that deals well with a data stream is a sliding-window approach. Effectively the sliding window is also a subsample of the data, existing of only the most recent data points. The advantages of this method are that memory burden is fixed and, in cases in which the data-generating process is not stationary over time, the most recent observations most heavily influence the resulting predictions. However, choosing the size of the window often requires domain knowledge: too small might not catch any events meaningful, too large a window might computationally be too expensive (see, Aggarwal, 2007, for an introduction on many more data-stream techniques, including sliding windows). In this chapter, we focus on another method to deal with data streams: *online learning*, "computing estimates of model parameters on-the-fly, without storing the data and by continuously updating the estimates as more observations become available" (Cappé, 2011b). Note that our current focus is solely on estimating the individual-level effects in the context of nested data and hence accounting for the grouping present in the data. While the inclusion of additional explanatory variables (for instance to take into account when an individual was last seen in the data stream, previous purchases, etc.) in the prediction model is possible when estimating shrinkage factors (see, for instance Morris and Lysy (2012) or Ippel, Kaptein, and Vermunt (2016b)), we restrict our attention solely to random-intercept models with binary outcomes.

A possible approach to efficiently obtain estimates in a situation where the data come streaming in, is to estimate the individual-level effects in real time using *online* estimated shrinkage factors. Online estimation (or online learning) implies that a parameter (e.g., a mean, or regression coefficient) is updated using a single (or small batch of) data point and some sufficient statistics (e.g., a summation of the previous data points, Bottou, 1999; Ippel, Kaptein, & Vermunt, 2016a). An illustrative example is the computation of the sample mean $\frac{1}{n}\sum_{t=1}^{n} x_t$. Estimating a sample mean in a

data stream using online learning can be done as follows:

$$n^{(t+1)} = n^{(t)} + 1$$
$$\bar{p}^{(t+1)} = \bar{p}^{(t)} + \frac{x^{(t+1)} - \bar{p}^{(t)}}{n^{(t+1)}},$$

or equivalently,

$$\begin{aligned} n &:= n + 1 \\ \bar{p} &:= \bar{p} + \frac{x - \bar{p}}{n}, \end{aligned} \tag{3.2}$$

where n is the total number of observations and ':=' is an assignment operator, meaning that the left-hand side is updated using the expression on the right-hand side. Throughout this chapter we will use the notation presented in Equation 3.2 as opposed to using explicit superscripts.

Note that the *offline* estimation procedure stores all the observations and for each new estimate revisits the older data points. Updating the sample mean offline in a data stream thus takes increasingly more time because more and more data need to be processed. On the contrary, the *online* estimation procedure only stores n and $\bar{p}$ in memory, and, when a new data point enters, these are updated according to Equation 3.2. This results in a time-constant update. Attractively, using online estimation methods, there is no need to revisit previous data points, which can therefore be discarded from memory (Kaptein, 2014). However, not every offline estimation procedure can be used exactly for online estimation (see, e.g., Ippel, Kaptein, & Vermunt, 2016a; Neal & Hinton, 1998). Hence, we often have to resort to approximate solutions. In this chapter, we evaluate the accuracy of online approximations of a number of shrinkage factors. Note that although we focus on data streams, extremely large static datasets can be analyzed using the same methods.

This chapter is organized as follows. Section 3.2 describes four existing shrinkage factors and develops the online implementation of each of the shrinkage factors. In Section 3.3, we discuss when the individual-level effect should be estimated, an issue which arises due to the fact that new data present themselves over time. Section 3.4 presents a simulation study where we compare the online and offline implementations of the shrinkage factors in terms of the accuracy of the estimated individual-level effects. Here we explicitly explore different data-generating mechanisms. In Section 3.5 we apply the developed online shrinkage factors to analyze a real dataset. The dataset contains data coming from a large panel study. Because dropouts in panel data is a serious threat, we focus on predicting the probability of non-response per repeatedly observed individual. These predictions could facilitate the choice of which respondents to invite for the next wave, or personalize the response request to achieve higher response rates. Finally, in Section 3.6, we discuss the limitations of the shrinkage factors and their possible extensions to a broader setting.

3.2 Estimation of shrinkage factors

The intuition of a shrinkage model (Eq. 3.1) is as follows: there is information available both on the group level as on the individual level, so by shrinking the individual-level effect towards the group mean, the estimator "borrows strength from the neighbors", thereby reducing the average squared prediction error (Efron & Morris, 1977; W. James & Stein, 1961; Stein, 1956). In this section, we discuss four shrinkage factors and develop their online implementations:

- James Stein estimator, (JS): Here, we use the formulation as introduced by Morris and Lysy (2012). This shrinkage factor assumes normally distributed individual-level effects. This assumption is clearly violated for binary data; using the data transformation, also suggested by Morris and Lysy (2012), the normal distribution is approximated. Furthermore, this shrinkage factor is equal across all individuals.
- Approximate Maximum Likelihood estimator, (ML): The ML is unlike the JS individual specific. The level of shrinkage is *influenced* by the number of observations of an individual. This shrinkage factor also assumes that the individual-level effects are normally distributed. Hence, here we also use the data transformation suggested by Morris and Lysy (2012).
- Beta Binomial estimator, (BB): This shrinkage factor does not assume a normal distribution, instead the individual-level effects are assumed to have a Beta distribution. Similar to ML, the level of shrinkage is individual specific and the level of shrinkage is influenced by the number of observations of an individual. We estimate the BB using the method of moments estimator (see, for instance Young-Xu & Chan, 2008).
- Heuristic estimator, (HN): Unlike the previous three shrinkage factors, the HN does not rely on any distributional assumptions. This shrinkage factor is an ad-hoc estimator which solely depends on the number of observations of an individual.

3.2.1 The James Stein estimator

The JS is historically important since it is among one of the first shrinkage factors to be considered in the literature. This shrinkage factor assumes normally distributed individual-level effects. Thus, the assumed data-generating model is:

$$\begin{aligned} \mathbf{y}_i &\sim N(\mu_i, \sigma_i^2 \mathbf{I}) \\ \mu_i &\sim N(\mu, \tau^2), \end{aligned} \tag{3.3}$$

where $\mathbf{y}_i$ is the response vector of individual i with n_i observations, $\mathbf{I}$ is a $n_i \times n_i$ identity matrix, σ_i^2 the residual variance, μ is the population average, which below

we estimated using $\bar{p}$, and τ^2 the variance of the individual-level effects. Since we focus on grouped binary data, the individual means (i.e., proportions) are bounded, and therefore, not nearly normally distributed. To address this, Morris and Lysy (2012) suggested the following data transformation:

$$w_i = \sqrt{\bar{n}}(\arcsin(1 - 2p_i) - \arcsin(1 - 2\bar{p})), \tag{3.4}$$

where w_i is the transformed individual mean, $\bar{n} = n/N$, the total number of observations divided by the total number of individuals, p_i the individual mean, $\bar{p}$ the sample mean over all data points. The transformation stabilizes the within-variance to be approximately equal to $\hat{\sigma}_i^2 = \bar{n}/n_i$. Using this data transformation to estimate the individual-level effects results in the following shrinkage model:

$$\hat{w}_i = w_i(1 - \hat{\beta}_{js}) + \bar{w}\hat{\beta}_{js}, \tag{3.5}$$

where $\bar{w}$ the average across the transformed individual means and $\hat{\beta}_{js}$, the JS shrinkage factor, is given by

$$\begin{aligned} \hat{\beta}_{js} =& \frac{N - 2}{\sum_{i=1}^{N} \frac{(w_i - \bar{w})^2}{\bar{n}/n_i}}, \\ =& \frac{N - 2}{SS_{js}}, \end{aligned} \tag{3.6}$$

as formulated by Morris and Lysy (2012), where SS_{js} is the sum of squares between individuals. To obtain the estimated individual-level effect in terms of probabilities one computes

$$\hat{\mu}_i = (\sin((\hat{w}_i/\bar{n}) + \arcsin(1 - 2\bar{p})) - 1)/-2. \tag{3.7}$$

Thus, the quantities or parameters that are needed to estimate μ_i using the JS shrinkage factor are: w_i, p_i, n_i, n, N, $\bar{p}$, $\bar{n}$, $\bar{w}$ and SS_{js} (see, Eq. 3.4, 3.5, and 3.6). While the above formulas detail how to estimate β_{js} in an *offline* setting, we now turn to deriving an *online* formulation.

Parts of the online computation of $\hat{\beta}_{js}$ are straightforward, for instance the computations of n or n_i using, $n := n + 1$, to merely count the observations. We do not detail these further. However, counting the number of unique individuals (N) requires some additional thought: before N is incremented when a new data point arrives, we need to check whether this new data point originates from an already observed individual or from a new individual. Only in the latter case we increment the counter:

$$N := \begin{cases} N & \text{if } i_t \in \mathbf{N}, \\ N + 1 & \text{if } i_t \notin \mathbf{N}, \end{cases} \tag{3.8}$$

where i is the index of an individual and subscript t indicates that we only focus on the individual belonging to the most recent data point. Furthermore, $\mathbf{N}$ is set of

unique identifiers of all known individuals observed up to now. Each individual is labeled with an identifier such that we can track the individual over time. If a new individual is observed a new element is added to the set $\mathbf{N}$. Thus, the vector of unique identifiers grows when *new* individuals arrive in the data stream, but does not grow when an *observed* individual arrives (again) in the data stream. To check whether the individual i_t is new or not, the set of unique identifiers of individuals $\mathbf{N}$ needs to be available.

The online update of the transformed individual means, $\bar{w}$, is less trivial than count observations or the online update of the sample mean (Eq. 3.2). The $\bar{w}$, is a sample mean averaged over individuals (N), not over data points (n). Similar to the count of individuals (N, Eq. 3.8), we check whether the individual belonging to the new data point is observed before. Different update functions are used depending on whether or not an individual is observed before. When the data point belongs to a known individual, there is already a contribution of this individual to $\bar{w}$. We, first, correct $\bar{w}$ by subtracting the old contribution (i.e, the previous w_i'), then, the new contribution (i.e., the updated w_i) is added to $\bar{w}$:

$$\bar{w} := \begin{cases} (N\bar{w} - w_i' + w_i)/N & \text{if } i_t \in \mathbf{N}, \\ (N\bar{w} + w_i)/N & \text{if } i_t \notin \mathbf{N}, \end{cases} \tag{3.9}$$

where w_i' is the previous transformed individual mean from the last time this individual (i_t) entered and w_i the current estimate of the transformed individual mean.

Note that, due to the influx of new data, group parameters ($n, \bar{p}$, and N) are constantly changing. The data transformation uses these group parameters. This implies that *all* transformed individual means change when new data enter, not only the individual that just entered (i_t). In order to obtain the *exact* same result using the offline and online estimation procedure, *all* the transformed individual means should be updated every time a data point enters. Updating all these transformed individual means is inefficient and becomes infeasible when the number of individuals grows rapidly. Hence, we approximate the offline version by updating only the current individual. We discuss this issue in more detail in Section 3.3.

The remaining parameter needed for the estimation of $\hat{\beta}_{js}$ is the between individuals sum of squares (SS_{js}), which is also a summation over individuals. For the estimation of SS_{js} we make use of a similar update regime as used for $\bar{w}$:

$$SS_{js} := \begin{cases} SS_{js} - SS_{js_{i'}} + \dfrac{(w_i - \bar{w})^2}{\bar{n}/n_i} & \text{if } i_t \in \mathbf{N}, \\ SS_{js} + \dfrac{(w_i - \bar{w})^2}{\bar{n}/n_i} & \text{if } i_t \notin \mathbf{N}, \end{cases} \tag{3.10}$$

where $SS_{js_{i'}}$ denotes the previous contribution to the SS_{js}. Using Eq. 3.10, β_{js} can be estimated, with which we can estimate $\hat{w}_i$ (Eq. 3.5). Lastly, to obtain $\hat{\mu}_i$, $\hat{w}_i$ is imputed in Eq. 3.7 to transform $\hat{w}_i$ to $\hat{\mu}_i$.

3.2.2 Approximate Maximum likelihood estimator

The ML is an often used shrinkage factor for multilevel models, where μ_i's are normally distributed and the outcome variable is continuous (among others, Goldstein, 1986). Because the means of binary observations are not normally distributed, we use the same data transformation (Eq. 3.4) as discussed previously in Section 3.2.1. Similar to the estimation of μ_i using the JS, the ML estimation of μ_i uses the alternative shrinkage model (Eq. 3.5) which includes the transformed individual means. However, unlike the previous shrinkage factor, ML estimator is tailored to an individual: the level of shrinkage is influenced both by the number of observations of an individual as well as by information of the other individuals:

$$\begin{aligned} \hat{\beta}_{ml\_i} &= \frac{\hat{\sigma}_i^2}{\hat{\tau}^2 + \hat{\sigma}_i^2}, \\ &= \frac{\bar{n}/n_i}{\hat{\tau}^2 + \bar{n}/n_i}, \end{aligned} \tag{3.11}$$

where more observations of an individual (n_i) result in less shrinkage, and $\hat{\tau}^2$ is the maximum-likelihood value of the variance of the individual-level effects. The most likely value of τ^2 is found by maximizing the following log-likelihood function (see, Morris & Lysy, 2012, equation at the bottom of page 128):

$$\begin{aligned} \ell(\tau^2) &= \sum_{i=1}^{N} \Big[\frac{(w_i - \bar{w})^2}{\bar{n}/n_i} \frac{\bar{n}/n_i}{\bar{n}/n_i + \tau^2} + \log\Big(\frac{\bar{n}/n_i}{\bar{n}/n_i + \tau^2} \Big) \Big] / 2, \\ &= \sum_{i=1}^{N} \Big[\frac{(w_i - \bar{w})^2}{\sigma_i^2} \frac{\sigma_i^2}{\sigma_i^2 + \tau^2} + \log\Big(\frac{\sigma_i^2}{\sigma_i^2 + \tau^2} \Big) \Big] / 2, \\ &= \sum_{i=1}^{N} \Big[\frac{(w_i - \bar{w})^2}{\sigma_i^2} \beta_{ml\_i} + \log(\beta_{ml\_i}) \Big] / 2, \end{aligned} \tag{3.12}$$

In the case of offline estimation, Eq. 3.12 is maximized by iterating over the dataset, using a numerical optimization method, for instance Newton Raphson.

For the estimation of μ_i using ML, the following parameters are needed: p_i, n_i, w_i, $\hat{\sigma}_i^2$, n, N, $\bar{n}$, $\bar{p}$, $\bar{w}$, and $\hat{\tau}^2$. Most of these parameters have already been discussed in the previous section (see, Eq. 3.2, 3.8, and 3.9), therefore we focus only on the remaining parameter: the estimation of the variance of the individual-level effects, $\hat{\tau}^2$.

Estimating $\hat{\tau}^2$ is not straightforward during the data stream since using an iterative maximization procedure is not feasible. For this reason, we use Stochastic Gradient Descent (SGD, Bottou, 2010). SGD updates the estimate of τ^2 by evaluating the gradient (in this case, a one-dimensional gradient or derivative) of $\ell(\tau^2)$ one data point at a time.

Intuitively, SGD works as follows: The first-order derivative of the log-likelihood function is a summation over individuals. SGD evaluates this first-order derivative

for a single data point and based on the value of the derivative SGD determines whether the current estimate of the parameter is above or below the maximum-likelihood value. Using a learn rate (γ), SGD steps towards the maximum of the likelihood function. When a new data point enters, SGD evaluates the derivative again and updates the parameter estimate accordingly. The first-order derivative of $\ell(\tau^2)$ is:

$$\nabla\ell(\tau^2) = \sum_{i=1}^{N} \frac{(w_i - \bar{w})^2 - \hat{\sigma}_i^2 - \hat{\tau}^2}{2(\hat{\sigma}_i^2 + \hat{\tau}^2)^2}. \tag{3.13}$$

Because Eq. 3.13 is a summation over individuals, we apply a similar update regime as in Equation 3.10:

$$\hat{\tau}^2 := \begin{cases} \hat{\tau}^2 - \gamma\nabla\ell_{i'}(\tau^2) + \gamma\nabla\ell_i(\tau^2) & \text{if } i_t \in \mathbf{N}, \\ \hat{\tau}^2 + \gamma\nabla\ell_i(\tau^2) & \text{if } i_t \notin \mathbf{N}, \end{cases}$$

where $\nabla\ell_{i'}(\tau^2)$ is the previous contribution of individual i to the gradient of τ^2 and $\nabla\ell_i$ the current contribution to that gradient of individual i. When the learn rate, γ, is large, SGD can 'move' fast towards the maximum-likelihood value, however with the same pace it can also step over the maximum of the likelihood function. When the learn rate is small it will take many evaluations of the derivative (i.e., many data points have to enter) before the maximum likelihood is reached (see, e.g., Bottou, 2010; Schaul, Zhang, & LeCun, 2013; Xu, 2011, for a more extensive discussion on learn rates for SGD). After the estimation of β_{ml_i}, the individual-level effect is estimated using the shrinkage model for transformed individual means (Eq. 3.5) after which $\hat{w}_i$ is transformed to $\hat{\mu}_i$ using Eq. 3.7.

3.2.3 The Beta Binomial estimator

When we assume that the data-generating model is a Beta Binomial distribution,

$$\begin{aligned} k_i &\sim Bin(n_i, \mu_i) \\ \mu_i &\sim Beta(\alpha, \beta) \end{aligned} \tag{3.14}$$

where $k_i = \sum_{j=1}^{n_i} y_{ij}$, the individual means do not have to be transformed to estimate BB, because the Beta distribution naturally falls within the $[0, 1]$ range. Thus, in order to estimate μ_i we can make use of the shrinkage model as defined in Eq. 3.1. In this case, we choose the method-of-moments estimation method to estimate BB because this method has a closed-form solution to estimate the shrinkage factor. The closed-form expression of the estimation procedure of BB makes it easier to rewrite the formulation of BB to an online formulation.

The compound distribution of the Beta Binomial distribution is:

$$f(k|n, \alpha, \beta) = \frac{\Gamma(n+1)}{\Gamma(k+1)\Gamma(n-k+1)} \frac{\Gamma(k+\alpha)\Gamma(n-k+\beta)}{\Gamma(n+\alpha+\beta)} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}, \tag{3.15}$$

where $k = \sum_{i=1}^{N} \sum_{j=1}^{n_i} y_{ij}$. Alternatively, the Beta Binomial distribution can also be parameterized as,

$$f(k|n, M, \mu) = \frac{\Gamma(M)}{\Gamma(M\mu)\Gamma(M(1-\mu))} \binom{n}{k} \frac{\Gamma(k + M\mu)\Gamma(n - k + M(1-\mu))}{\Gamma(n + M)} \quad (3.16)$$

where $\mu = \frac{\alpha}{\alpha+\beta}$, which estimated by $\bar{p}$, and $M = \alpha+\beta$, which is computed as follows:

$$\hat{M} = \frac{\bar{p}(1-\bar{p}) - \hat{s}^2}{\hat{s}^2 - \frac{\bar{p}(1-\bar{p})}{N} c}.$$

Here, $c = \sum_{i=1}^{N} 1/n_i$ and $\hat{s}^2$ is defined as

$$\begin{aligned} \hat{s}^2 =& \frac{N \sum_{i=1}^{N} n_i (p_i - \bar{p})^2}{(N-1) \sum_{i=1}^{N} n_i}, \\ =& \frac{N SS_{bb}}{(N-1) \sum_{i=1}^{N} n_i}, \end{aligned}$$

where SS_{bb} is the between-individual sum of squares using the individual means. The shrinkage factor of BB is:

$$\hat{\beta}_{bb\_i} = \frac{\hat{M}}{\hat{M} + n_i}, \quad (3.17)$$

Similar to the ML, the BB is also individual specific where the number of observations per individual influences the level of shrinkage. The parameters for the estimation of μ_i using BB are: n_i, p_i, SS_{bb_i}, n, N, c, $\bar{p}$, $\hat{M}$, and $\hat{s}^2$.

The computation of $\hat{s}^2$ requires the following: N, n, and SS_{bb}. While the first two parameters are easy to update during the data stream and already discussed in Section 3.2.1, the latter is again a sum over individuals, which requires an update similar to SS_{js} (Eq. 3.10):

$$SS_{bb} := \begin{cases} SS_{bb} - SS_{bb_{i'}} + n_i(p_i - \bar{p})^2 & \text{if } i_t \in \mathbf{N}, \\ SS_{bb} + n_i(p_i - \bar{p})^2 & \text{if } i_t \notin \mathbf{N}, \end{cases}$$

where $SS_{bb_{i'}}$ denotes the previous contribution to the SS_{bb}. Similar to the $\hat{\beta}_{js}$, the $\hat{\beta}_{bb}$ estimated online is slightly different compared to the offline estimated $\hat{\beta}_{bb}$. The difference between the two estimation procedures is due to the fact that SS_{bb} is dependent on $\bar{p}$ which fluctuates throughout the data stream.

For the computation of $\hat{M}$ we need $\bar{p}, \hat{s}^2, N$, and c. Because all parameters except the last parameter are already discussed previously, we only present the computation of c:

$$c := \begin{cases} c - c_{i'} + 1/n_i & \text{if } i_t \in \mathbf{N}, \\ c + 1/n_i & \text{if } i_t \notin \mathbf{N}, \end{cases}$$

where $c_{i'}$ is the previous contribution to c (i.e. $\frac{1}{n_i - 1}$). The individual-level effect μ_i is then estimated using Eq. 3.1, using $\hat{\beta}_{bb\_i}$, p_i and $\bar{p}$.

Table 3.1: Overview of the characteristics of the shrinkage factors and their complexity

	JS	ML	BB	HN
distribution μ_i	Normal	Normal	Beta	–
group or individual	group	individual	individual	individual
equal variance	yes	no	no	no
transformation	yes	yes	no	no
update $\hat{\mu}_i$	p_i, n_i, w_i, SS_{js_i}	p_i, n_i, w_i, $\hat{\tau}_i^2$, $\hat{\sigma}_i^2$	p_i, n_i, SS_{bb_i}	p_i, n_i
	$\bar{p}$, $\mathbf{N}$, n, N	$\bar{p}$, $\mathbf{N}$, n, N	$\bar{p}$, $\mathbf{N}$, n, N	$\bar{p}$, $\mathbf{N}$, n
	$\bar{n}$, $\bar{w}$, SS_{js}	$\bar{n}$, $\bar{w}$, $\hat{\tau}^2$	SS_{bb}, $\hat{s}^2$, c, $\hat{M}$	–

3.2.4 The Heuristic estimator

The previous two shrinkage factors (ML, Eq. 3.11 and BB, Eq. 3.17) both have a similar type of intuition: individual-level effects are moved more towards the group mean when little is known about the individual (i.e., a small number of observations) compared to when there is more information about an individual. The last shrinkage factor has the same intuition, however, we do so without any distributional assumptions or sophisticated formulas. The last shrinkage factor

$$\hat{\beta}_{hn\_i} = \frac{1}{\sqrt{n_i}},$$

shrinks individual-level effects only based on the (square root of) number of observations of an individual. Like BB, the HN also shrinks the individual-level effects using Eq. 3.1: When an individual only has 1 observation, $\hat{\mu}_i = \bar{p}$, and the amount of shrinkage decreases as more observations of an individual enter. All the parameters used for the estimation of μ_i using HN (p_i, n_i, $\bar{p}$, and n), have been discussed in Section 3.2.1.

Table 3.1 gives an overview of the online shrinkage factors that are used in the simulation study. The characteristics of each of the shrinkage factors are presented. The last three lines of the table give an indication how many parameters estimates should be updated to estimate the shrinkage factor and the individual-level effect when an additional data point enters the dataset. First of the three lines are the individual parameters, second line are group level count parameters, and last line are the parameters that require more computations to update.

3.3 Predicting individual-level effects: when is the right time?

When analyzing static data, the exact moment at which one should predict the individual-level effects, does not come to question. It naturally follows that the prediction is only made once: after the shrinkage factor is estimated. This is, however, not the case when data are entering over time. In this section, we explain why the researcher is faced with a choice when to estimate the individual-level effect.

The individual-level effect is a combination of the individual mean, the group mean, and a shrinkage factor. Every time a new data point enters, the record of one person changes. However, due to this new data point, the estimates of all the individual-level effects at that moment in time change slightly. That is, *if* one would re-estimate all the individual-level effects every time a new data point enters, the estimates change with every additional data point. Such re-estimation is, however, infeasible for the full set of individuals at each time point, and in many applications one would obtain an estimate only for the individual concerned. In any case, the shrinkage of the individual-level mean to the group-level mean to obtain a prediction for a specific individual can be done at two distinct moments:

1. one could obtain a shrinkage estimate the moment an individual's data is observed and store the resulting prediction,

2. or, one could obtain a prediction at the moment that the individual is about to re-enter the dataset; hence, the moment a prediction might be needed.

The first option leads to the following procedure: when a data point enters, the group-level parameters, the parameters of the individual (i_t) whose data point entered, and shrinkage factor are updated or computed. With these parameters, a prediction of the individual-level effect is made and stored in memory. This option has two downsides. The first downside is that besides p_i, a prediction ($\hat{\mu}_i$) needs to be stored, which is potentially never used if we do not observe this individual anymore. The other downside is that while we store the prediction, new data are entering. These new data points affect the shrinkage factor and global statistics. All these changes are not taken into account because the prediction is stored and considered fixed. Therefore, the stored prediction does not optimally make use of the most recent information.

For the second option, imagine an individual (i_t) intends to pay our website—as discussed in the introduction—a visit again. Her browser will send out a request to access the website. At that point, we know who is about to enter our website, so we can retrieve this individual's record. Now, we can predict this individual's $\hat{\mu}_i$ based on all the information we know so far. The data generated by this individual during the website visit allows us to update both the group and individual-level parameters estimates. This second option thus deals with both downsides of the first option: it does not waste memory on storing predictions that we might end up not using at all and it incorporates the most recent changes to the group-level parameters.

The two options are illustrated in Figure 3.1. The black dot denotes an individual mean. One could choose to predict the individual-level effect right after this data point enters at $t = 1$, or one could wait until this individual returns ($t = 2$) and shrink towards the group mean at that point in time (which is $t = 2$ in this case). As can be seen from the plot, *when* the individual-level effect is estimated influences the estimate of μ_i. Because the group mean and the estimated shrinkage factor change over time, these two options (shrink at $t = 1$ vs. $t = 2$) do not result in the same

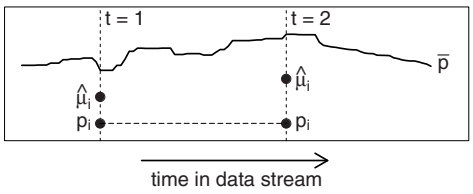


Figure 3.1: An illustration of when to shrink the individual-level effect. Option 1 ($t = 1$) estimates μ_i right after the data point enters, option 2 estimates μ_i at $t = 2$. While p_i remains the same between the two time points, the group mean $\bar{p}$, does change over time.

$\hat{\mu}_i$. In the following simulation study, we have chosen this second implementation of predicting the individual-level effect.

3.4 Simulation Study

3.4.1 Design

To evaluate whether the online implementations of the shrinkage factors perform equally well as their original offline implementations, we conduct a simulation study. In this simulation study we compare the two estimation procedures in terms of the average squared prediction error ($\sum(\hat{\mu}_i - \mu_i)^2/N$). Since two of the four shrinkage factors assume a normal distribution, we specifically examine the case when this is violated in the simulation study. To do so, we generated three distributions of the individual-level effects, which increasingly depart from normality: the distribution underlying the *true* individual-level effects is centered $B(7, 7)$, right skewed, $B(2, 12)$, or a mixture of two Beta distributions: $B(1, 6)$ and $B(6, 1)$. We set the average[2] number of observations equal to 20. As a benchmark we use a multilevel model with logit link function, as implemented in the GLMR function from the 'lme4' package (in [R]) with 20 adaptive Gaussian Quadrature points. While this model is known to provide very good estimates of μ_i, it is computationally complex to fit (Agresti, Booth, Hobert, & Caffo, 2000; Bock & Aitkin, 1981; Breslow & Clayton, 1993; Moerbeek, Van Breukelen, & Berger, 2003; Rabe-Hesketh, Skrondal, & Pickles, 2002; Skrondal & Rabe-Hesketh, 2004), especially in a data stream. The generated data streams consist of $n = 10,000$ (which results in $N = 500$) and all conditions have 1,000 replications.

[2]Because we sample the individuals at random after which we generate an observation, the number of observations is not equal across individuals.

3.4.2 Results

The main results of the simulation study are presented in Figure 3.2 and Figure 3.3. Figure 3.2 presents the average of the estimated shrinkage factors over the simulation runs. Figure 3.3 presents the average squared prediction error over the simulation runs. Both figures consist of three subfigures: one for the centered ($B(7,7)$ distribution, one for the right skewed ($B(2,12)$) distribution, and one for the mixture distribution ($B(1,6)$ and $B(6,1)$). Table 3.2 further details the average squared prediction error at three points in the data stream and includes the standard deviation over the different simulation runs.

The x-axis of Figure 3.2 presents the length of the data stream and the y-axis the average shrinkage factor. The solid lines represent the online implementations of the shrinkage factors. The dashed lines represent the offline implementations of the shrinkage factors. The four gray lines indicate the offline (dashed) and the online (solid) shrinkage factors that do not require the data transformation. The BB carries triangle symbols (facing up) to differentiate the BB from HN which carries square symbols. The black lines are also marked with symbols: JS is denoted with circles and ML is marked with triangles (facing down). In all three subfigures, there is a small difference between the offline and online implementations of the shrinkage factors. In general, the online implementations tend to shrink somewhat more than the offline implementations.

In Figure 3.2a the centered distribution is presented. The BB (online and offline) shrinks the individual-level effects most, and the online implementation does so even more than the offline implementation. The BB (online and offline) needs many (over 2,000) data points before it can be estimated. This is an artifact of the method of moments estimator, which returns negative hyperparameters for the Beta distribution when the data does not (yet) comply to the beta distribution (under dispersion). Both the offline versions of the JS and the ML have a relatively stable level of shrinkage, while the online implementation of the JS quickly decreases during the data stream. The ML online implementation only changes very gradually. The chosen learn rate ($\gamma = 0.01$) might have been slightly too small. Towards the end of the generated data streams three of the four shrinkage factors shrink approximately the same, only the heuristic shrinkage factor (online and offline) shrinks substantially less than the other factors.

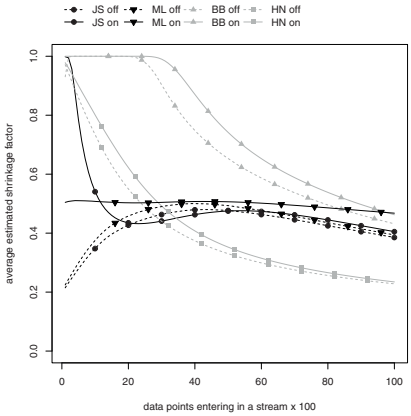
The average estimated shrinkage factors in the right-skewed distribution of the individual-level effects are presented in Figure 3.2b. For the two shrinkage factors that do not use the data transformation the results are quite similar. However, the ML and JS show differences with the previous condition. The online implementation of the JS shrinks more over a longer time, also the offline implementation of the JS shrinks more in the beginning of the data stream. The offline ML shrinks on average some more than the offline JS but behaves qualitatively the same as the offline JS. Towards the end of the data stream, the different shrinkage factors are more spread

out than in the previous condition, while the online and offline implementations of all four shrinkage factors have similar levels of shrinkage.

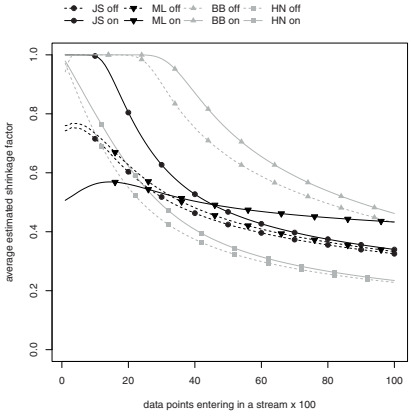
The last subfigure (Fig. 3.2c) presents a different pattern of shrinkage factors. Even at the end of the data stream, there are two distinct clusters of shrinkage factors. The cluster of shrinkage factors with the highest level of shrinkage consists of the online and offline implementations of the heuristic shrinkage factors, and the online implementation of ML. The remaining shrinkage factors (online and offline BB and JS, and the offline ML) hardly shrink at all. This is due to the fact that the data-generating distribution of the individual-level effects is bimodal. Because the heuristic estimator (online and offline) does not have any distributional assumptions, it cannot take into account that there are two modes. The online ML does decrease more in this condition than in the other two conditions. A larger learn rate or longer data stream would allow the online ML to decrease even more and reach a similar level of shrinkage as the offline ML. The offline ML, BB and JS do take into account that the distribution of individual level effects is not normal, and shrink very little accordingly.

The subfigures of Figure 3.3 are organized as follows: The y-axes present the average squared prediction error: $\sum(\hat{\mu}_i - \mu_i)^2/N$ and the x-axes present the data stream. Note that the y-axes of the three subfigures of Figure 3.3 differ across the three scenarios. In addition to the already introduced lines, the dotted line represents the GLMR function. The results of the two unimodal distributions ($B(7,7)$ and $B(2,12)$) are comparable, however, the mixture distribution ($B(1,6), B(6,1)$) shows different results. Figure 3.3a and Figure 3.3b show that in the beginning of the data stream, the two shrinkage factors that make use of the data transformation have more error (JS, ML) than the two shrinkage factors (BB, HN) that do not rely on the transformation. The GLMR function performs 'best' in both scenarios. However, the difference between the shrinkage factors and the GLMR function is minimal later in the data stream. More importantly for our purpose, there is almost no difference between the *offline* and *online* implementations of the shrinkage factors.

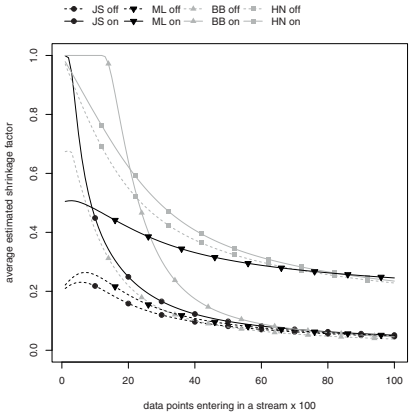
Table 3.2 is organized as follows. In the rows are the three conditions (centered, right skewed and mixture), within each condition three points within the data stream are presented (n = 1,000, 5,000, and 10,000). In the columns are the different shrinkage factors with the offline and online implementations. Both the average squared prediction error of each of the shrinkage factors and the standard deviations are presented. In the centered scenario, the GLMR function outperforms the shrinkage factors (offline and online). However, as the data stream continues, the difference between the shrinkage factors and GLMR becomes smaller. The standard deviations across the shrinkage factors and during the stream are stable and small. The second scenario, the right-skewed distribution, has an even smaller average squared prediction error. This is due to the fact that the distribution of μ_i is narrowly distributed around the group mean making the mean over all data a good



(a) $B(7, 7)$

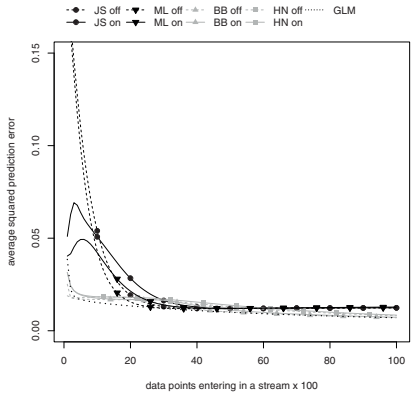


(b) $B(2, 12)$

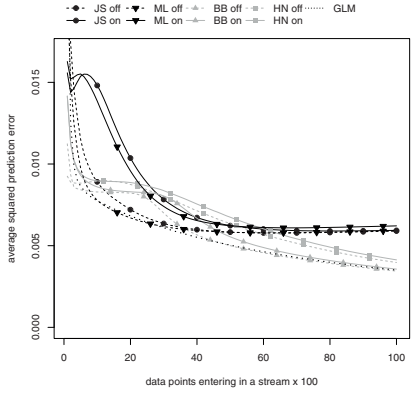


(c) $B(1, 6)$ & $B(6, 1)$

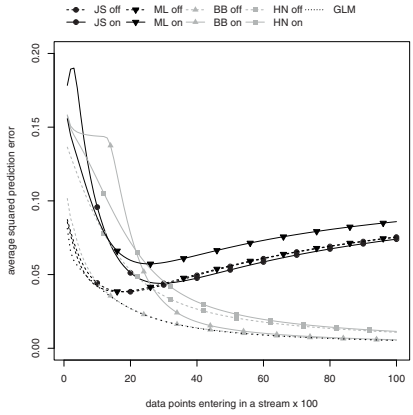
Figure 3.2: The average estimated shrinkage factors, averaged over the 1,000 replications



(a) $B(7, 7)$



(b) $B(2, 12)$



(c) $B(1, 6)$ & $B(6, 1)$

Figure 3.3: Average squared prediction error for the three scenarios, averaged over the 1,000 replications.

predictor of the individual-level effects. This results in a small average squared prediction error and even smaller standard deviations.

The mixture scenario provides quite different results. While the average squared prediction error decreases rapidly in the beginning of the data stream (see Fig. 3.3c), after about 2,000 data points the error *increases* for both JS and ML. For the other two shrinkage factors and the benchmark GLMR, i.e., these estimation methods that do not use Morris and Lysy's (2012) data transformation, this is not the case. This pattern appears for both the online and offline estimated shrinkage factors. Due to the mixture of the two distributions the individual means are either clustered close to zero, or close to one. While the mean of these two distributions is 0.5, all the true individual-level effects are either close to zero or close to one. This makes the group mean a poor predictor of the individual-level effects. Because these individual means are far from the group mean, the transformed individual means have large absolute values. In an absolute sense, larger values are moved more towards the group mean than values that are closer to the group mean. Transforming the predicted individual-level effects to $\hat{\mu}_i$'s causes the individual-level effects to be moved even more towards the group mean. Thus, even though the shrinkage factors that use the data transformation are in fact small (see Figure 3.2c), the data transformation pushes the individual-level effects even closer to the group mean. This additional push towards the group mean causes the JS and ML to have larger prediction error than HN and BB.

From the simulation study, we can thus conclude that a) for a long enough data stream all online shrinkage factors perform as well as their offline counterparts, and b) the BB seems to have the most robust performance over the three conditions. Hence, for the analysis of large, nested, binary outcome data streams we would recommend using the our online version of the BB. In the following section, all the examined shrinkage factors are further evaluated in a real-data example. In this example we show that it is possible to accurately predict whether respondents of a long-running panel study will respond to a monthly questionnaire.

3.5 LISS Panel Study: Predicting Attrition

An application where data are entering over time and real-time prediction is relevant is a panel study, where new questionnaires are sent to the same respondents over a longer period of time. Panel studies are used to analyze ongoing trends. One of the major issues of a panel study is attrition (i.e., respondents who drop out) because it can affect the generalizability of the results to the population (Goodman & Blum, 1996). Much effort is spent on the prevention of non-response, for instance, reminders, rewards (Curtin, Singer, & Presser, 2007; Manzo & Burke, 2012), and multimode data collection (Leeuw, 2005). Knowing which respondents are likely to drop out of the panel, could facilitate the prevention of the dropout. For instance, when the probability for a given respondent to answer to the questionnaire drops below

Table 3.2: The average squared prediction error. In the parentheses are the sd's over 1,000 replications.

		JS				ML				BB				HN				GML	
		offline		online		offline		online		offline		online		offline		online		offline	
distribution	n	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)	$\bar{e^2}$	(sd)
	1,000	.054	(.004)	.051	(.004)	.043	(.004)	.042	(.004)	.017	(.001)	.018	(.001)	.018	(.001)	.018	(.001)	.015	(.001)
$B(7,7)$	5,000	.012	(.001)	.012	(.001)	.012	(.001)	.012	(.001)	.010	(.001)	.011	(.001)	.013	(.001)	.014	(.001)	.010	(.001)
	10,000	.012	(.001)	.012	(.001)	.012	(.001)	.013	(.001)	.007	(.000)	.007	(.000)	.008	(.001)	.008	(.001)	.007	(.000)
	1,000	.009	(.001)	.015	(.001)	.008	(.001)	.014	(.001)	.008	(.001)	.009	(.001)	.009	(.001)	.009	(.001)	.008	(.001)
$B(2,12)$	5,000	.006	(.000)	.006	(.000)	.006	(.000)	.006	(.000)	.005	(.000)	.006	(.001)	.006	(.000)	.007	(.001)	.005	(.000)
	10,000	.006	(.000)	.006	(.000)	.006	(.000)	.006	(.000)	.003	(.000)	.004	(.000)	.004	(.000)	.004	(.000)	.004	(.000)
	1,000	.044	(.005)	.096	(.006)	.042	(.004)	.088	(.005)	.045	(.005)	.144	(.003)	.086	(.003)	.115	(.003)	.042	(.004)
$B(1,6), B(6,1)$	5,000	.056	(.001)	.053	(.002)	.056	(.001)	.068	(.002)	.011	(.001)	.013	(.001)	.021	(.001)	.024	(.001)	.011	(.001)
	10,000	.075	(.002)	.074	(.002)	.076	(.002)	.086	(.002)	.005	(.000)	.006	(.001)	.011	(.001)	.011	(.001)	.005	(.000)

a threshold, an additional incentive (a letter of the importance of the panel, money etc.) could be sent when inviting the respondent to answer the questionnaire to increase the probability that the respondent will reply to the questionnaire. Knowing a respondent was unlikely to respond to the questionnaire, after the facts, is not very informative or helpful. Therefore, predicting non-response in a panel study is a good example of a case where real-time prediction is useful.

In this application, we predict whether a respondent of the panel study is going to participate in the next wave as well. Data are coming from the LISS (Longitudinal Internet Study for Social sciences) panel study, consisting of 50 monthly waves between November 2007 and December 2011. For each wave, respondents either scored a 1 if they participated in that wave or a 0 if they failed to participate. For the analysis, we selected only these respondents that received at least one questionnaire, who had an identification number and started before December 2011. Total number of individuals used for the analysis is $N = 12,924$ and the total number of observations, $n = 397.647$. For the analysis of the LISS panel data, we had to drop one questionnaire (July, 2011) because none of the respondents had answered this questionnaire.

We analyze the data by replaying the data as if it is a data stream. To do so, we kept the ordering of the questionnaires but randomly ordered the respondents within a questionnaire. We randomly selected the responses within a questionnaire because we do not have data about the order in which the data entered originally. We compare the results of the four shrinkage factors (online and offline) with the results by the GLMR function, like in the simulation study, in terms how well each of the methods can classify whether a respondent is or is not going to respond. We take into account when a respondent stopped being a member of the panel[3] and correct the group statistics accordingly.

3.5.1 Results

Figure 3.4 presents the results of the replay of the data stream of the LISS panel questionnaires. The y-axis presents the percentage of correctly classified respondents. A respondent is correctly classified if the shrinkage model predicted the probability of a response greater than .5 and the respondent indeed answered the questionnaire, or when the predicted probability was below .5 and the responded failed to answer the questionnaire. The x-axis is the replay of the questionnaires as these are send out over time.

As expected from the simulation study, the differences between the offline and online estimation procedures are negligible. The classification performances of the offline BB and GLMR are exactly the same, and therefore, impossible to disentangle. Furthermore, the same clustering of shrinkage factors as in the simulation condition with the mixture of distribution appears in Figure 3.4: the JS and ML (online and

[3] An extra variable in the data set gives (when applicable) the date the responded stopped being a member of the panel. After which the respondent is no longer invited for the next waves.

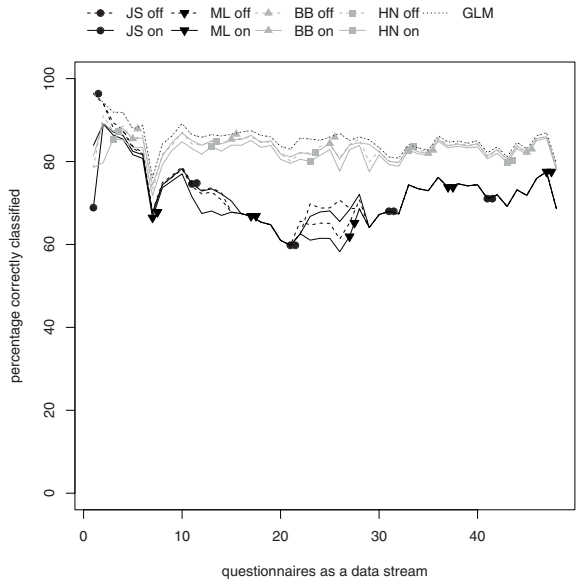


Figure 3.4: Percentage correctly classified responses

offline) are less able to make accurate predictions with regard to non-response while the HN and BB perform equally well as the benchmark (GLMR). This is not much of a surprise, as the distribution of the individual-level effects estimated with GLMR (the MAP estimates) shows that the majority of the individual-level effects are on either end of the interval, see Fig. 3.5 with not many respondents in the middle, just like the mixture of distributions of the simulation study. Even though BB and HN are less computationally complex than GLMR, the predictions made by BB and HN are equally accurate.

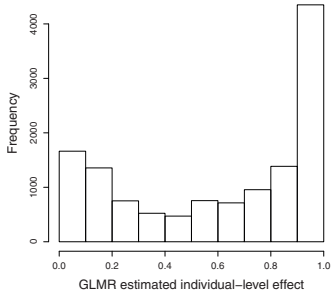


Figure 3.5: Estimated μ_i using the GLMR function.

3.6 Conclusion and discussion

The most important conclusion we can draw is that we can make accurate predictions of the individual-level effects when the outcome variable is binary, the data have a nested structure, when the data enter over time, and predictions are required in real time. While the multilevel model with logit link function is the standard for analyzing nested data with a binary outcome, due to the computational complexity of that model, analyzing data streams of nested binary data becomes infeasible. To overcome this problem, we studied online – and computationally efficient – versions of four different shrinkage factors: the James Stein estimator, the (approximate) Maximum Likelihood estimator, the Beta Binomial estimator and lastly a heuristic estimator. In a simulation study, we showed that all these shrinkage factors on average make good predictions of the individual-level effects. However when there is a mixture of distributions of the individual-level effects, shrinkage factors that do not rely on the normal distribution of the individual-level effects do noticeably better. It appears that the data transformation suggested by Morris and Lysy (2012), in the studied situations does not work well in situations where the number of observations is limited and the distribution of the individual-level effects deviates from the normal distribution.

There are differences between the shrinkage factors in how well they are able to predict the individual-level effects. When the true individual-level effects are close to normally distributed the prediction accuracy is very similar across all shrinkage factors. More importantly, the shrinkage factors implemented offline (making use of all individual data points) or online (incrementally and not revisiting previous data) perform similarly. However, when the distribution of the individual-level effects deviates from the normal distribution, the James Stein (JS) estimator and the approximate Maximum Likelihood estimator perform less well than the Beta Binomial and heuristic estimator.

In the current study, we assumed the data-generating process to be stationary; the possible effect of the time within the data stream is not explicitly modeled. As a result, the individual-level effects are estimated using the information of all data points equally, irrespective of their history. In practice, this assumption might, however, not hold. If the stationarity assumption does not hold, one might prefer to weigh the recent data points more heavily than the older data points when computing an estimate. All the online shrinkage factors presented in this chapter are easily adapted to create such a moving window approach by changing the learn rate of the procedure to a fixed value: for example, when updating the sample mean $\bar{p}$ using Equation 3.2 we effectively use a learn rate of $\frac{1}{n}$ (which is easy to see since $\frac{x-\bar{p}}{n} = \frac{1}{n}(x - \bar{p})$). By choosing a fixed learn rate of, e.g., $\frac{1}{1000}$ instead we would (smoothly) decrease the value of older data points in the resulting estimate.

A possible advantage of the JS and ML could be that these methods are easier to extend to deal with covariates (see, for instance, Morris & Lysy, 2012). The JS

can easily include fixed effects to improve the prediction of the individual effect. Including more random effects in this case might be less straightforward. The ML can, however, facilitate more random effects as well as fixed effects (Ippel, Kaptein, & Vermunt, 2016b) at the level of the group. Including fixed effects at the level of the observations seems challenging for both the JS as well as the ML since the suggested data transformation aggregates the information to the level of individuals.

Making real-time predictions without revisiting older data has great potential. These real-time predictions are not only beneficial in the context of e-commerce, or to encourage respondents that have a low probability to respond to the questionnaire. Other cases include classifying credit-card transactions (legitimate versus fraudulence transactions), monitoring patients' compliance with their medical regimen (medication was taken or not), or tracking students' progress in their educational career (passing exams or not), to name a few. The presented methods for estimating individual-level effects in data streams allow the researcher to take into account the dependence among the observations without losing the computational efficiency of the methods that do not take this dependency into account.

Acknowledgement

We would like to thank prof.dr. Peter Lugtig for preparing and sharing the LISS panel dataset. Additionally we would like to thank dr. Marcel Croon for his continuous help and feedback on this chapter.

Chapter 4

Estimating Random-Intercept Models on Data Streams

Abstract

Multilevel models are often used for the analysis of grouped data. Grouped data occur for instance when estimating the performance of pupils nested within schools or analyzing multiple observations nested within individuals. Currently, multilevel models are mostly fit to static datasets. However, recent technological advances in the measurement of social phenomena have led to data arriving in a continuous fashion (i.e., data streams). In these situations the data collection is never "finished". Traditional methods of fitting multilevel models are ill-suited for the analysis of data streams because of their computational complexity. A novel algorithm for estimating random-intercept models is introduced. The Streaming EM Approximation (SEMA) algorithm is a fully-online (row-by-row) method enabling computationally-efficient estimation of random-intercept models. SEMA is tested in two simulation studies, and applied to longitudinal data regarding individuals' happiness collected continuously using smart phones. SEMA shows competitive statistical performance to existing static approaches, but with large computational benefits. The introduction of this method allows researchers to broaden the scope of their research, by using data streams.

This chapter is published as Ippel, L., Kaptein, M.C, & Vermunt, J.K. (2016) Estimating Random-Intercept Models on Data Streams. *Computational Statistics and Data Analysis*, *104*, 169–182

4.1 Introduction

In the social sciences, we often encounter grouped data, such as pupils grouped within school classes (e.g., P. Barrett, Zhang, Moffat, & Kobbacy, 2013), multiple observations grouped within individuals (Killingsworth & Gilbert, 2010), or voters grouped within geographical regions (Gelman, 2007). Such data are typically analyzed using multilevel (or hierarchical) models in which batches of group-level statistics are treated as randomly drawn from an underlying distribution. In this chapter, we will use the formulation of "observations nested within individuals", although the method we present does not restrict itself to this type of nesting.

Multilevel models have various advantages over more traditional methods of analysis, such as aggregated analysis, in which the within-group structure is ignored, or group-specific analysis, in which information about the other groups is ignored. That is, they

1. contain fewer parameters than group-specific models,
2. allow for generalization of results to a wider population of groups, and
3. allow information to be shared between groups (Raudenbush & Bryk, 2002; Steenbergen & Jones, 2002).

The latter property in particular makes multilevel analysis interesting when the focus is on obtaining group-level predictions, since multilevel modeling yields smaller out-of-sample prediction error than predictions derived from either an aggregate or a group-specific analysis (see e.g., Morris & Lysy, 2012).

Current (maximum-likelihood) methods for fitting multilevel models use iterative algorithms such as Newton-Raphson or Expectation Maximization (EM, Dempster, Laird, & Rubin, 1977) to maximize the likelihood. Alternatively, but not considered in this chapter, one could use a Bayesian framework with MCMC sampling (for more details see, e.g., Browne & Goldstein, 2010). However, each of these methods require multiple passes through the full dataset to obtain parameter estimates. Even though fitting a multilevel model once, on a moderately sized dataset does often not require excessive computation time, such ways of fitting multilevel models can become infeasible when a dataset is extremely large, or in the situation where the data collection is never "finished" because more data present themselves over time.

Recent technological developments have, however, led to the increased availability of these so-called data streams: i.e., datasets which are continuously augmented with new data points. Such data streams often have a grouped (or nested) structure. Examples include fraud detection using credit card transactions, where transactions are nested within credit cards (Patidar & Sharma, 2011), telephone communication analysis, where calls are nested within telephone registrations (Cortes, Fisher, Pregibon, Rogers, & Smith, 2000), and consumer behavior tracking in e-commerce, where purchased items or visited web pages are nested within customers (Lee, Podlaseck, Schonberg, & Hoch, 2001). In order to obtain up-to-date predictions of the

individual-level effects, the estimates of the parameters of the model of interest should be updated as data points come in, and the updated estimates of the model parameters should be used for prediction purposes. When applied to streaming data, these traditional methods have to repeatedly cycle through all available data points, each time a new data point arrives, in order to obtain up-to-date parameter estimates. Additionally, even if the dataset is no longer augmented, but static and (extremely) large, it is often computationally preferable to analyze the dataset in smaller batches, or even a data point at a time (Ng & McLachlan, 2003; Thiesson, Meek, & Heckerman, 2001). We propose an adaption of the EM algorithm for the estimation of random-intercept models, to resolve the problem of analyzing grouped data in a data stream or when the dataset is extremely large.

The resulting Streaming EM Approximation algorithm (henceforth referred to as SEMA) falls within the framework of online learning methods (Gaber et al., 2005). A key feature of online learning is that the data are summarized into a few summary statistics which contain all relevant information of previous data points (Opper, 1998). SEMA is an approximate EM method, because unlike the EM algorithm which uses all the data to update the estimates of the model parameters, we only use a single data point, some summary statistics on the individual level, and the previous estimates of the model parameters, to update the estimates of the model parameters. Because SEMA does not require all the data to be in memory, SEMA is more appropriate to deal with data streams than the conventional EM algorithm.

Related methods for speeding up the EM algorithm have been proposed for dealing with large (static) datasets, for example, Berlinet and Roland (2012) discussed methods to speed up the convergence rate of the conventional EM algorithm. Wolfe, Haghighi, and Klein (2008) presented an (offline) parallel version of the EM algorithm and McLachlan and Peel (2000, ch. 12) described various possible adaptations of EM methods for large datasets. Various online adaptations of the EM algorithm for different applications have also been proposed, for example, for mixture models (see, e.g., Liu, Almhana, Choulakian, and McGorman, 2006; McLachlan and Peel, 2000; Wolfe et al., 2008) and for latent variable models (Cappé & Moulines, 2009). Instead of speeding up the EM algorithm, Steiner and Hudec (2007) proposed a method to scale down the data prior to using the EM algorithm. We add to this existing literature by proposing an EM approximation for the estimation of models based on data streams consisting of *dependent* observations. The method we propose stores information on the level of individuals, instead of the level of observations, and updates the estimates in a single pass over the data, making it suitable for both data streams and extremely large datasets.

The remainder of this chapter is organized as follows. In the next section, we illustrate the computational advantages of streaming estimation using the simple example of the estimation of a sample mean. Next, we discuss the estimation of random-intercept models using the EM algorithm, and show how this algorithm

can be modified into a streaming version, leading to SEMA. Subsequently, we evaluate SEMA in two simulation studies. In the first simulation study, we evaluate the accuracy of the estimates of the model parameters, and of the individual-level effects. In the second study, we evaluate three alternative implementations of SEMA to improve the estimates both of the model parameters and of the individual-level effects. The first alternative uses a small part of the data to obtain better starting values, the second implementation cycles through all individuals at given intervals, and the last implementation is a combination of the previous two. In Section 4.5, we illustrate the use of SEMA in an application using real data on respondents' happiness, in which nested data, collected using a smart-phone application, "arrived" in a stream. In Section 4.6, we detail some theoretical characteristics of SEMA, and we discuss a convergence diagnostic to evaluate the estimated model parameters of SEMA. In Section 4.7, we extend the random-intercept model to include additional fixed covariates. The last section discusses the main results of the simulation studies and presents directions for future work.

4.2 From offline to online data analysis

Before introducing SEMA, we first explain the key changes involved when moving from the *offline* analysis of static datasets to the *online* analysis of data streams. This conceptual shift is easily illustrated by examining the computation of a sample mean $\bar{x}_n$. The standard offline computation proceeds as follows:

$$\bar{x}_n = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{4.1}$$

where x_i denotes the measurement for the ith unit and n the total number of observations.

Suppose now that we want to compute the sample mean *and* that data enter in a stream. The naive application of the above offline formula would then imply that each time a new data point enters one has to count the number of observations n and compute the sum of all measurements x_i. This is feasible as long as n is not too large or when the update is only required rarely. However, even a simple computation as in Equation 4.1 becomes infeasible when it needs to be performed in the face of rapidly entering data points, as n grows larger and larger.

The online computation of a sample mean can be done by noting that the sample mean for $n + 1$ data points can be expressed as an *update* of the estimated sample

mean for n data points $x_1, \ldots, x_n$. More specifically,

$$\begin{aligned} \bar{x}_{n+1} &= \frac{\sum_{i=1}^{n+1} x_i}{n+1}, \\ &= \bar{x}_n \frac{n}{n+1} + \frac{x_{n+1}}{n+1}, \\ &= \bar{x}_n + \frac{x_{n+1} - \bar{x}_n}{n+1}. \end{aligned} \tag{4.2}$$

The last line of Equation 4.2 shows two key features of online learning: first, when a new observation enters, we update the current estimate without revisiting all the historical data. This reduces the computational complexity required to update the sample mean. Note that the number of (offline) computations needed to compute the sample mean as n grows, progresses as follows:

$$\begin{aligned} 1 + 2 + 3 + \cdots + n &= \frac{1}{2} n(n+1), \\ &= O(n^2). \end{aligned}$$

In comparison, the online update of the sample mean, requires the following number of computations

$$\begin{aligned} 1 + 1 + 1 + \cdots + 1 &= n, \\ &= O(n). \end{aligned}$$

This simple analysis shows that the computations to update the mean offline grow quadratically in n, while online the number grows linearly as a function of n.

Second, only certain summary statistics (here n and $\bar{x}_n$) are kept in memory. This makes online learning both computationally fast as well as memory efficient. Similar algorithms can be used, amongst others, for updating of higher moments (Welford, 1962) or for estimating the coefficients of a linear regression model using least squares (Escobar & Moser, 1993; Plackett, 1950). In the next section, we detail the transition from offline estimation to online estimation of the random-intercept model.

4.3 Online estimation of random-intercept models

4.3.1 The random-intercept model and its standard offline estimation

In this section, we describe the random-intercept model with continuous outcomes, which we focus on throughout this chapter. Next, we give a conceptual description of the EM algorithm, as well as the technical details of fitting the random-intercept model. These technical details are subsequently needed to explain the transition from offline estimation of the random-intercept model to the online estimation of this model.

The model of interest can be formulated as follows:

$$y_{ij} = \mu_j + \epsilon_{ij}, \quad i = 1 \dots n_j,\ j = 1 \dots J, \tag{4.3}$$

where y_{ij} is observation i of individual j, n_j is the total number of observations of an individual, J is the total number of individuals, and μ_j is the individual-level random intercept. These intercepts are assumed to be normally distributed as $\mu_j \sim \mathcal{N}(\mu, \tau^2)$. The random error per observation is denoted by ϵ_{ij} and is also assumed to be normally distributed $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ and independent of μ_j. The three unknown model parameters to be estimated are thus μ, τ^2, and σ^2.

Maximum-likelihood estimates of the parameters of the random-intercept model cannot be computed directly due to the fact that μ_j is not observed. In order to obtain maximum-likelihood estimates, we use the Expectation Maximization algorithm (Dempster et al., 1977). The EM algorithm uses the complete-data log-likelihood function, a likelihood function for which the latent variable (μ_j) is assumed to be known. The complete-data log-likelihood function is as follows:

$$\begin{aligned} \ell(\mu, \tau^2, \sigma^2 | y) = &-\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\sigma^2 - \frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\frac{(y_{ij}-\mu_j)^2}{\sigma^2} \\ &-\frac{J}{2}\ln(2\pi) - \frac{J}{2}\ln\tau^2 - \frac{1}{2}\sum_{j=1}^{J}\frac{(\mu_j-\mu)^2}{\tau^2}, \end{aligned} \tag{4.4}$$

where $n = \sum_{j=1}^{J} n_j$.

Because μ_j is not observed, we have to impute values for this variable in order to compute the Complete Data Sufficient Statistics (CDSS). There are three CDSS, one for each model parameter. We denote these CDSS for μ, τ^2, and σ^2 by T_1, T_2, and T_3, respectively. In order to compute the CDSS, the algorithm imputes values for the latent variable in the E step. Using these imputed values in combination with the estimates of the model parameters of the previous iteration (or starting values) the CDSS are computed. Subsequently, these CDSS are used in the M step. The M step maximizes the complete-data log-likelihood (Eq. 4.4), given the CDSS of the previous E step.

The CDSS computed in the E step are a function of three individual-level statistics. These individual-level statistics are a function of the observations of individual j and the estimates of the model parameter at iteration $k-1$. These individual-level statistics are $\hat{\mu}_{j(k)}$, $\hat{\rho}_{j(k)}$, and $\hat{\nu}_{j(k)}$, which represent the individual-level effect, the reliability of this individual-level effect, and variance of this individual-level effect respectively, at iteration k.

We can obtain, $\hat{\mu}_{j(k)}$ using

$$\hat{\mu}_{j(k)} = \hat{\rho}_{j(k)}\bar{y}_j + (1 - \hat{\rho}_{j(k)})\hat{\mu}_{(k-1)}, \tag{4.5}$$

where $\bar{y}_j$ is the individual average, and where $\hat{\rho}_j$ equals

$$\hat{\rho}_{j(k)} = \frac{\hat{\tau}^2_{(k-1)}}{\hat{\tau}^2_{(k-1)} + \hat{\sigma}^2_{(k-1)}/n_j}. \tag{4.6}$$

Note that one minus the reliability, $\hat{\rho}_{j(k)}$, can be interpreted as a *shrinkage factor*, which determines the extent to which the estimated individual-level effect, $\hat{\mu}_j$, is moved towards the overall mean, $\hat{\mu}$, (see for instance, Morris & Lysy, 2012; Stein, 1956). When τ^2 is large compared to the residual variance, σ^2, the reliability goes up. The reliability also goes up when the number of observations per individual, n_j, increases. Lastly, we compute

$$\hat{\nu}_{j(k)} = \hat{\tau}^2_{(k-1)}(1 - \hat{\rho}_{j(k)}), \tag{4.7}$$

which can be interpreted as a measure of uncertainty of the individual-level effect.

The CDSS are then computed as follows:

$$T_{1(k)} = \sum_{j=1}^{J} \hat{\mu}_{j(k)}, \tag{4.8}$$

$$T_{2(k)} = \sum_{j=1}^{J} (\hat{\mu}^2_{j(k)} + \hat{\nu}_{j(k)}), \tag{4.9}$$

$$T_{3(k)} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} \left[(y_{ij} - \hat{\mu}_{j(k)})^2 + \hat{\nu}_{j(k)} \right]. \tag{4.10}$$

In the M step, these CDSS are used to obtain new estimates $\hat{\mu}_{(k)}$, $\hat{\tau}^2_{(k)}$, and $\hat{\sigma}^2_{(k)}$. That is,

$$\hat{\mu}_{(k)} = \frac{T_{1(k)}}{J}, \tag{4.11}$$

$$\hat{\tau}^2_{(k)} = \frac{T_{2(k)}}{J} - \hat{\mu}^2_{(k)}, \tag{4.12}$$

$$\hat{\sigma}^2_{(k)} = \frac{T_{3(k)}}{n}. \tag{4.13}$$

After updating the estimates of the model parameters a new E step is executed, followed by an M step. This process is repeated until convergence.

4.3.2 Online estimation of the random-intercept model

For streaming estimation of the random-intercept model, an algorithm is needed that does not require storing all the data in memory, or cycling through all the data points at each iteration cycle. For this purpose, we propose a modification of the E step of the EM algorithm described previously. This modification involves updating the contribution to the CDSS *only* for the individual for which a new data point enters.

The M step remains the same since, given the CDSS, the M step is independent of the data points.

The key feature used by our proposed SEMA algorithm is that the CDSS T_1, T_2, and T_3 can be computed at the level of individuals instead of observations within individuals. Therefore, it is no longer required to store all n observations, we merely store a small number of summaries for each of the J individuals.

Let j_t denote the individual which corresponds to the t-th data point. Note that this data point can be either from an individual who is already in the sample, or from a new individual. For the discussion of SEMA, the iteration index, which was previously denoted by k, is now replaced by t, indexing the data point which is being processed. The key element of the proposed SEMA algorithm is that $\hat{\mu}_j$, $\hat{\rho}_j$, and $\hat{\nu}_j$ are computed only for individual $j = j_t$, that is, the individual for which a new data point arrives. This implies that when going from the CDSS based on $t-1$ data points, denoted by $T_{w(t-1)}$, $w \in \{1, 2, 3\}$ to those based on t data points, $T_{w(t)}$, *only* the contribution of individual $j = j_t$ needs to be updated. This can be expressed as follows:

$$T_{w(t)} = T_{w(t-1)} - T_{wj_t(t-1)} + T_{wj_t(t)}, \tag{4.14}$$

where $T_{wj_t(t-1)}$ and $T_{wj_t(t)}$ denote the contribution to CDSS for individual j_t before and after the entry of data point t. Note that $T_{wj(t)} = T_{wj(t-1)}$ for $j \neq j_t$; that is, the contribution does not change if the new data point does not concern individual j.

Equation 4.8 (T_1, CDSS for μ) and Equation 4.9 (T_2, CDSS for τ^2) are already written as a sums over J individuals instead of data points. Therefore they are easily rewritten in the format of Equation 4.14:

$$\begin{aligned} T_{1(t)} &= \sum_{j=1}^{J} \hat{\mu}_{j(t)}, \\ &= \sum_{j=1}^{J} T_{1j(t)}, \\ &= T_{1(t-1)} - T_{1j_t(t-1)} + T_{1j_t(t)}. \end{aligned} \tag{4.15}$$

The difference between Equation 4.8 and Equation 4.15 is that in the former all $\hat{\mu}_j$ are estimated with the estimates of the model parameters from the latest iteration. In the latter formulation, however, only for person j_t, $\hat{\mu}_j$ is computed using the most recent estimates of the model parameters. Therefore, SEMA applies a *partial* E step, only for 1 individual (see also, McLachlan & Peel, 2000; Neal & Hinton, 1998). We

rewrite T_2 in a similar way:

$$\begin{aligned} T_{2(t)} &= \sum_{j=1}^{J} \hat{\mu}^2_{j(t)} + \hat{\nu}_{j(t)}, \\ &= \sum_{j=1}^{J} T_{2j(t)}, \\ &= T_{2(t-1)} - T_{2j_t(t-1)} + T_{2j_t(t)}. \end{aligned} \tag{4.16}$$

The update of the CDSS of the residual variance (T_3, Eq. 4.10) differs from the previous two equations. This is due to the fact that Equation 4.10 presents a summation over n observations, while here we present the computation more efficiently as a summation over J individuals. However, this is relatively straightforward:

$$\begin{aligned} T_{3(t)} &= \sum_{j=1}^{J} \sum_{i=1}^{n_j} \left[(y_{ij} - \hat{\mu}_{j(t)})^2 + \hat{\nu}_{j(t)} \right], \\ &= \sum_{j=1}^{J} (\bar{y^2_j} - 2\bar{y}_j \hat{\mu}_{j(t)} + \hat{\mu}^2_{j(t)} + \hat{\nu}_{j(t)}) n_j, \\ &= \sum_{j=1}^{J} T_{3j(t)}, \\ &= T_{3(t-1)} - T_{3j_t(t-1)} + T_{3j_t(t)}, \end{aligned} \tag{4.17}$$

where $\bar{y^2_j}$ is the average of the squared y_{ij} for individual j. This analysis shows that in order to perform the E step we do not need all data points, but only $\bar{y}_j$, $\bar{y^2_j}$ and n_j.

To summarize, at entry of data point t the SEMA algorithm proceeds as follows:

- E step: for $j = j_t$,
  1. subtract the current contribution from the CDSS,
  2. update $\bar{y}_j$, $\bar{y^2_j}$, and n_j online,
  3. compute new $\hat{\mu}_{j(t)}$, $\hat{\rho}_{j(t)}$, and $\hat{\nu}_{j(t)}$,
  4. add the new contribution to the CDSS,
- M step
  1. increase J by 1 when it concerns an observation of a new individual and set $n = t$,
  2. compute the new estimates $\hat{\mu}_{(t)}$, $\hat{\tau}^2_{(t)}$, and $\hat{\sigma}^2_{(t)}$ based on the CDSS from the previous E step.

In the summary presented above, it can be seen why SEMA (Streaming EM Approximation) is called an approximate EM algorithm: SEMA performs, like EM, an E step and a M step. However, unlike the EM algorithm, it only does a single partial E step, because it only updates the contribution to the CDSS for a single individual.

Doing only an E step for one individual is computationally less expensive than doing the E step for all individuals. It also means that SEMA will converge more slowly (i.e., take more partial E - M steps) to the (local) maximum-likelihood estimate than the EM algorithm, because it only updates the information of one individual instead of the updating the information of all individuals. The benefit of SEMA is that it is computationally less intensive, which makes it suitable for dealing with both very large static data and data streams, because the required memory only grows with the number of individuals instead of the number of data points. An example of the SEMA algorithm in [R] code is available at `http://github.com/L-Ippel/SEMA`. In Section 4.6, we provide some additional justification for SEMA. In the next section, we will test the accuracy of SEMA in two simulation studies.

4.4 Performance of SEMA evaluated by simulation

4.4.1 Simulation study I: Evaluation of the precision of estimated parameters

Design

In this simulation study, we compare the performance of the proposed SEMA algorithm with the standard EM algorithm, in terms of the accuracy of the parameter estimates. An important factor affecting the speed of convergence of the EM algorithm for multilevel models is the average reliability $\bar{\rho}$ (see also Eq. 4.6); that is, when $\bar{\rho}$ is large, the EM algorithm will converge after a few iterations, but when the ρ_j's get closer to zero convergence will become slower. Since it can be expected that convergence of SEMA will be strongly affected by $\bar{\rho}$, this is the main factor varied in this simulation study. We do so in two different ways: by varying the number of observations per individual, n_j, and by varying the amount of variance of the random intercept, τ^2. In this simulation study, we keep the residual variance, σ^2, constant. We evaluate SEMA and EM by monitoring the parameter estimates and predicted individual-level effects during the data stream.

We generated data streams of $n = 10,000$ observations. The average number of observations per individual (n_j) was 10, 25, or 100, which results in $J =$1,000, 250, or 100 individuals in total. The individual-level effects, μ_j, were drawn from a normal distribution with $\mu = 10$ and variance $\tau^2 = \{1, 10, 25, 100\}$. The residual variance was set to $\sigma^2 = 100$ in all conditions. First, we generated J individual-level effects from $\mu_j \sim \mathcal{N}(\mu, \tau^2)$. Next, the observations were generated by randomly drawing an individual, and generating a data point based on this individual's true individual-level effect. The 12 different settings for n_j and τ^2 yielded average reliabilities $\bar{\rho}$ ranging from .091 to .990. Table 4.1 presents the different levels of $\bar{\rho}$ in the simulation study.

Each of these 12 conditions was run 1,000 times. The starting values were $\hat{\mu}_{(0)} = y_{t=1}$ (first observation of the data stream), $\hat{\tau}^2_{(0)} = 1$, and $\hat{\sigma}^2_{(0)} = 1$. Both the simulation

of the data stream, and the estimation using SEMA and EM, were implemented in [R](R Core Team, 2013).

Results

Tables 4.2 through 4.5 present the mean and standard deviation (SD) of $\hat{\mu}$, $\hat{\sigma}^2$, and $\hat{\tau}^2$ respectively, and the averaged squared prediction error: $\bar{e^2} = \sum_{j=1}^{J} \frac{(\hat{\mu}_j - \mu_j)^2}{J}$ across 1,000 replications at 100, 1,000, 5,000, and 10,000 observations for both SEMA and standard (offline) EM. For each simulation run, the population values were $\mu = 10$ and $\sigma^2 = 100$. The two factors that varied are τ^2 (presented in the columns) and n_j (presented in the rows). In bold are the parameter estimates that differed by more than 10 compared to the population values that generated the data.

Across all conditions, the SEMA and EM estimates of μ are close to the population value even with as little as 100 observations (see Table 4.2). However, the SEMA estimates are clearly much more variable than those of EM at the beginning of the data stream, that is, when SEMA did not have the chance to converge. But this difference has disappeared by 10,000 observations. Both EM and SEMA have larger SD's at the end of the data stream in the condition with $\tau^2 = 100$ and $n_j = 100$ than in the other conditions. This is due to the smaller number of individuals in this condition ($J = 100$), causing the data-generating model to fluctuate more across the different runs.

The results for the estimated residual variance, σ^2, are presented in Table 4.3. In the condition in which τ^2 is low, both EM and SEMA underestimate the residual variance in the beginning of the data stream (when $n = 100$). SEMA somewhat overestimates σ^2 halfway through the data stream for the conditions where $\tau^2 > 1$. In all conditions, the variability of both the SEMA and EM estimate is large in the beginning of the data stream, but decreases steadily towards the end of the data stream. Irrespective of the average reliability, $\bar{\rho}$, at the end of the data stream ($n = 10,000$), σ^2 is estimated equally well by SEMA and EM.

Next, Table 4.4 presents the results for τ^2. In the five lowest reliability conditions ($\tau^2 = 1$ with $n_j = \{10, 25, 100\}$ and $\tau^2 = 10$ with $n_j = \{10, 25\}$, see Table 4.1), SEMA seems to slightly overestimate τ^2. However, when $n =$ 5,000, SEMA starts approaching the EM estimates. The only situation in which τ^2 still seems overestimated at 10,000 observations occurs with $\tau^2 = 1$ and $n_j = 10$, which is the lowest

Table 4.1: Average reliability $\bar{\rho}$ in the simulation study for $\sigma^2 = 100$

	τ^2			
n_j	1	10	25	100
10	.091	.500	.714	.909
25	.200	.714	.862	.962
100	.500	.909	.962	.991

Table 4.2: estimates of μ (population value $\mu = 10$) averaged over 1,000 replications ($\sigma^2 = 100$). In the parentheses are the SD's over 1,000 replications.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	10.10	(3.90)	9.99	(1.05)	10.08	(4.13)	9.99	(1.09)	10.10	(4.46)	9.99	(1.16)	10.10	(5.92)	9.99	(1.46)
	1,000	10.07	(2.67)	10.00	(0.32)	10.06	(2.81)	10.00	(0.34)	10.08	(3.04)	10.00	(0.38)	10.12	(4.09)	10.00	(0.52)
	5,000	10.01	(0.89)	10.00	(0.15)	10.01	(0.92)	10.00	(0.17)	10.01	(0.97)	10.00	(0.21)	10.03	(1.07)	10.00	(0.34)
	10,000	10.00	(0.22)	10.00	(0.10)	10.00	(0.19)	10.00	(0.14)	10.00	(0.19)	10.00	(0.18)	10.00	(0.32)	10.00	(0.32)
25	100	9.82	(3.95)	10.06	(1.05)	9.84	(4.14)	10.07	(1.06)	9.85	(4.42)	10.07	(1.13)	9.91	(5.81)	10.10	(1.43)
	1,000	9.91	(2.01)	10.00	(0.32)	9.93	(2.11)	10.01	(0.37)	9.93	(2.22)	10.01	(0.43)	9.96	(2.98)	10.02	(0.63)
	5,000	10.00	(0.17)	10.00	(0.15)	10.00	(0.22)	10.01	(0.22)	10.01	(0.29)	10.01	(0.29)	10.01	(0.52)	10.01	(0.52)
	10,000	10.00	(0.12)	10.00	(0.12)	10.01	(0.19)	10.01	(0.19)	10.01	(0.27)	10.01	(0.27)	10.01	(0.51)	10.01	(0.51)
100	100	9.93	(3.56)	9.98	(1.03)	9.95	(3.73)	10.01	(1.12)	9.99	(4.03)	10.03	(1.25)	10.09	(5.23)	10.09	(1.70)
	1,000	10.01	(0.93)	10.01	(0.34)	10.04	(1.05)	10.03	(0.47)	10.08	(1.24)	10.05	(0.62)	10.15	(1.76)	10.10	(1.09)
	5,000	10.01	(0.18)	10.01	(0.18)	10.03	(0.35)	10.03	(0.35)	10.04	(0.53)	10.04	(0.53)	10.09	(1.04)	10.09	(1.04)
	10,000	10.01	(0.15)	10.01	(0.15)	10.03	(0.34)	10.03	(0.34)	10.05	(0.53)	10.05	(0.53)	10.10	(1.01)	10.10	(1.04)

Table 4.3: Estimates of σ^2 averaged over 1,000 replications, ($\mu = 10, \sigma^2 = 100$). In the parentheses are the SD's over 1,000 replications and in bold those values which are more than 10 from the true value.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	98.23	(35.04)	**74.87**	(30.82)	107.40	(38.39)	**78.11**	(33.73)	**122.24**	(44.24)	**82.52**	(38.26)	**198.37**	(73.47)	95.12	(55.96)
	1,000	96.47	(20.26)	97.91	(5.31)	103.18	(22.55)	100.00	(6.60)	**114.29**	(26.49)	100.27	(7.13)	**170.38**	(49.62)	100.27	(7.46)
	5,000	96.61	(3.42)	99.85	(2.16)	99.60	(4.54)	100.08	(2.27)	103.01	(6.51)	100.08	(2.27)	109.48	(15.04)	100.08	(2.28)
	10,000	98.08	(1.48)	100.00	(1.48)	99.79	(1.56)	100.03	(1.50)	100.23	(1.57)	100.03	(1.50)	100.13	(1.51)	100.02	(1.50)
25	100	98.85	(33.41)	**85.61**	(21.79)	107.50	(36.36)	**88.58**	(24.74)	**121.39**	(40.72)	92.33	(28.93)	**193.94**	(72.38)	99.56	(40.14)
	1,000	96.33	(11.60)	98.75	(4.87)	101.03	(14.15)	99.89	(5.57)	107.98	(17.70)	99.90	(5.67)	**139.70**	(40.37)	99.87	(5.77)
	5,000	98.47	(2.03)	99.92	(2.08)	99.84	(2.10)	99.96	(2.10)	100.00	(2.11)	99.96	(2.10)	99.97	(2.10)	99.96	(2.10)
	10,000	99.51	(1.41)	99.93	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)	99.94	(1.44)
100	100	98.40	(29.94)	92.77	(15.72)	105.21	(34.93)	95.48	(17.65)	**116.66**	(42.86)	97.68	(20.03)	**173.49**	(72.73)	99.00	(22.93)
	1,000	98.71	(16.66)	99.65	(4.62)	100.70	(20.88)	100.00	(4.80)	101.70	(25.59)	100.00	(4.80)	103.33	(42.81)	100.00	(4.81)
	5,000	99.95	(2.05)	99.99	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)	100.00	(2.06)
	10,000	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)	99.96	(1.45)

reliability condition corresponding to $\bar{\rho} = 0.091$. In general, the higher the reliability the faster the estimate of τ^2 converges to its true value.

The last result we present from this simulation study is the average squared prediction error of the individual-level effects, $\bar{e^2}$. Table 4.5 shows that irrespective of the condition, $\bar{e^2}$ and its variability across replications are very large in beginning of the data stream for both SEMA and EM, but both the size and the variability of $\bar{e^2}$ decrease rapidly during the data stream; that is, when the model parameter estimates improve and the amount of information available per individual increases. The prediction quality of SEMA is similar to that of EM at 5,000 data points, except for the lowest reliability condition ($\bar{\rho} = 0.091$) in which SEMA performs somewhat worse. When the reliability increases, as expected, the prediction error decreases for both SEMA and EM. For a stream of length $n = 10,000$ the performance of EM and SEMA is identical.

4.4.2 Simulation study II: Improving SEMA in low reliability cases

Design

Our first simulation study showed that in the lowest reliability condition, i.e., when $n_j = 10$ and $\tau^2 = 1$, SEMA performs less well than EM. That is, the average estimates of SEMA for τ^2 are too high (at $n = 10,000$: SEMA: $\hat{\tau}^2 = 5.47$, EM: $\hat{\tau}^2 = 1.04$) and for σ^2 are too low (at $n = 10,000$: SEMA: $\hat{\sigma}^2 = 98.08$, EM: $\hat{\sigma}^2 = 100.00$) and moreover, the average squared prediction error, $\bar{e^2}$, of SEMA ($\bar{e^2} = 1.92$) is larger compared to EM ($\bar{e^2} = 0.94$).

One possible explanation for the fact that SEMA has some difficulties in the low reliability condition is that it is sensitive to the starting values, especially when the average reliability is very low. Our rather crude starting values may have been too far off to guarantee convergence within 10,000 data points. A second explanation is that this low reliability condition is a rather difficult condition even for EM. That is, in a situation where the average reliability equals .091, the EM algorithm needs hundreds of iterations (and passes through the full dataset) to converge. It is not surprising that the SEMA algorithm, which passes through the dataset only once, has not yet reached the peak of the likelihood function.

These two explanations suggest two possible adaptations of SEMA: a) an adaptation yielding better starting values and b) an adaptation in which more than one pass over all individuals is performed. For this purpose, we investigate three possible variants of the SEMA algorithm, which we refer to as SEMA-T, SEMA-U, and SEMA-TU, where the T refers to *training*, and the U to *update*. That is,

1. **SEMA-T**: While SEMA is used to obtain estimates of the individual-level effect and the model parameters, when the *first* 1,000 observations of the data stream have entered, the EM algorithm (which iterates until convergence) is used to obtain better estimates for the model parameters, which are subsequently used for SEMA.

Table 4.4: Estimates of τ^2 averaged over 1,000 replications, ($\mu = 10, \sigma^2 = 100$). In the parentheses are the SD's over 1,000 replications and in bold those values which are more than 10 from the true value.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	**18.76**	(11.55)	**26.11**	(28.80)	**20.33**	(12.51)	**31.75**	(32.27)	23.13	(14.46)	**42.12**	(37.63)	**36.24**	(24.22)	103.40	(59.83)
	1,000	**17.07**	(8.88)	3.40	(2.66)	18.99	(10.02)	10.20	(5.10)	22.35	(12.17)	24.86	(6.56)	**39.02**	(23.78)	99.62	(10.78)
	5,000	10.47	(3.22)	1.21	(0.68)	14.19	(4.41)	9.98	(1.37)	22.08	(6.90)	24.98	(2.08)	**78.85**	(20.20)	100.01	(5.53)
	10,000	5.47	(0.87)	1.04	(0.47)	11.00	(1.53)	10.01	(0.90)	24.32	(2.24)	25.02	(1.58)	99.42	(5.09)	100.05	(5.01)
25	100	**19.62**	(12.77)	**14.39**	(18.10)	**21.45**	(14.46)	**20.49**	(21.92)	24.42	(16.73)	31.67	(27.77)	**38.22**	(27.44)	98.68	(47.36)
	1,000	**15.13**	(6.87)	2.00	(1.83)	17.79	(8.32)	9.82	(3.79)	22.68	(10.81)	24.75	(5.13)	**51.44**	(27.11)	99.46	(11.28)
	5,000	4.20	(0.57)	1.00	(0.58)	10.40	(1.37)	9.92	(1.29)	24.68	(2.47)	24.85	(2.39)	99.42	(7.90)	99.51	(7.88)
	10,000	1.80	(0.22)	1.00	(0.35)	9.95	(1.03)	9.95	(1.03)	24.88	(2.14)	24.88	(2.14)	99.51	(7.64)	99.51	(7.64)
100	100	**17.61**	(11.34)	7.17	(9.47)	19.50	(12.68)	13.23	(13.42)	22.96	(15.19)	25.70	(18.85)	**40.22**	(28.04)	98.00	(34.68)
	1,000	5.43	(1.35)	1.20	(1.23)	10.80	(2.95)	9.72	(2.93)	23.61	(5.85)	24.54	(5.05)	96.73	(18.83)	98.67	(15.63)
	5,000	1.05	(0.37)	0.98	(0.42)	9.88	(1.72)	9.87	(1.72)	24.71	(3.86)	24.71	(3.86)	98.86	(14.52)	98.85	(14.52)
	10,000	0.98	(0.29)	0.98	(0.29)	9.88	(1.57)	9.88	(1.57)	24.71	(3.70)	24.71	(3.70)	98.86	(14.34)	98.86	(14.34)

Table 4.5: The average squared error ($\bar{e^2} = \sum_{j=1}^{J} (\hat{\mu}_j - \mu_j)^2 / J$) averaged over 1,000 replications. In the parentheses are the SD's over 1,000 replications.

		population values of τ^2															
		$\tau^2 = 1$				$\tau^2 = 10$				$\tau^2 = 25$				$\tau^2 = 100$			
		SEMA		EM		SEMA		EM		SEMA		EM		SEMA		EM	
$\bar{n}_j$	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
10	100	20.43	(27.19)	16.43	(23.88)	28.06	(29.93)	23.44	(22.19)	40.80	(34.65)	33.62	(19.48)	106.75	(63.51)	65.08	(16.99)
	1,000	11.73	(16.96)	1.26	(0.40)	18.26	(18.89)	9.05	(0.67)	29.18	(22.12)	18.66	(1.23)	82.87	(43.91)	42.16	(2.62)
	5,000	4.16	(2.62)	0.99	(0.06)	8.35	(3.45)	6.84	(0.32)	13.70	(5.02)	11.80	(0.57)	26.10	(13.32)	19.33	(0.98)
	10,000	1.92	(0.37)	0.94	(0.05)	5.24	(0.29)	5.15	(0.24)	7.63	(0.40)	7.56	(0.35)	10.08	(0.53)	10.01	(0.48)
25	100	21.12	(26.19)	7.16	(11.35)	28.49	(28.55)	15.14	(10.81)	40.55	(32.72)	26.30	(10.09)	103.92	(63.21)	56.48	(13.67)
	1,000	8.58	(9.76)	1.17	(0.28)	13.96	(11.79)	8.20	(0.68)	22.06	(14.71)	15.69	(1.20)	57.81	(36.65)	31.05	(2.36)
	5,000	1.53	(0.21)	0.93	(0.08)	4.59	(0.34)	4.57	(0.33)	6.36	(0.47)	6.35	(0.47)	7.98	(0.60)	7.98	(0.60)
	10,000	0.89	(0.08)	0.82	(0.06)	2.92	(0.21)	2.92	(0.21)	3.56	(0.26)	3.56	(0.26)	4.00	(0.29)	4.00	(0.29)
100	100	17.89	(26.17)	3.57	(3.79)	24.63	(30.97)	11.46	(3.99)	36.01	(38.36)	21.70	(5.02)	90.48	(66.36)	45.24	(9.68)
	1,000	2.74	(15.29)	1.10	(0.27)	6.29	(19.35)	5.30	(0.80)	9.13	(23.92)	7.63	(1.13)	13.17	(40.94)	9.98	(1.50)
	5,000	0.70	(0.11)	0.71	(0.11)	1.70	(0.25)	1.70	(0.25)	1.89	(0.27)	1.89	(0.27)	2.01	(0.29)	2.01	(0.29)
	10,000	0.52	(0.08)	0.52	(0.08)	0.92	(0.13)	0.92	(0.13)	0.98	(0.14)	0.98	(0.14)	1.01	(0.15)	1.01	(0.15)

2. **SEMA-U**: A single EM iteration over all available individuals is used to update all the estimated individual-level effects and model parameters after *each* 1,000 data points.

3. **SEMA-TU**: combines both features.

The training set could provide SEMA with better starting values, speeding up the convergence to a local maximum. The second variant of SEMA, using EM updates is especially useful when observations of an individual enter in a block. In that case, the contributions to the CDSS will be based on estimates of the model parameters which are not yet converged, and more importantly these erroneous contributions to the CDSS are not corrected, because this individual is no longer returning. Doing an additional full E step will help in correcting the contributions to the CDSS. In this second simulation study, we repeat the $n_j = 10$ and $\tau^2 = 1$ condition but now we also apply these three variants of SEMA. Additionally we keep track of the computational time required by each of the different algorithms: EM, SEMA, and the three variants of SEMA.

Results

Table 4.6 presents the results obtained with the different variants of SEMA at $n =$ 900, 1,000, 5,000, and 10,000 observations. At 900 observations, all SEMA versions are still identical, but at 1,000 observations large differences appear between the variants using those observations as a training set and those that do not. For μ, the average estimates were already close to the true value at $n = 900$ observations, but clear improvements are visible in the SDs, with the variants of SEMA with a training set having lower SDs than those without a training set. At 5,000 observations, the difference between EM and SEMA-T and SEMA-TU are minimal. The training set in SEMA-T and SEMA-TU clearly improves the precision of the estimates of μ, while the additional update only marginally improves the precision.

However, for τ^2, the SEMA variants have a large impact on both the point estimate and their SD. Allowing for a single EM updates every 1,000 data points (SEMA-U) already yields a solution that is much closer to the full EM solution. An even larger improvement is shown by SEMA-T. Using both a training set and EM updates yields another slight improvement of the estimate of τ^2. A similar pattern can be observed for the residual variance σ^2, though the effect is smaller because the SEMA estimate was already close to the true value. Using a training set and EM updates yields estimates closer to those of the EM algorithm, though the additional updates seem to only have a minimal influence on the estimate and its SD.

The average squared prediction error is more affected by using a training set or additional EM updates. This effect of the training set and updates is especially noticeable halfway through the data stream. The variant with only the training dataset outperforms the variant with only the updates. Towards the end of the data stream, the difference between standard SEMA and its variants becomes much smaller.

Table 4.6: Results of SEMA variants in the condition $\mu = 10$, $\tau^2 = 1$, and $\sigma^2 = 100$. In the parentheses are the SD's over 1,000 replications, and in bold those values which are more than 10 from the population value: $\tau^2 = 1$.

		SEMA		SEMA T		SEMA U		SEMA T+U		EM	
	n	mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
$\hat{\mu}$	900	10.07	(2.74)	10.07	(2.74)	10.07	(2.74)	10.07	(2.74)	10.00	(0.33)
	1,000	10.07	(2.67)	10.00	(0.32)	10.06	(2.32)	10.00	(0.32)	10.00	(0.32)
	5,000	10.01	(0.89)	10.00	(0.24)	10.00	(0.41)	10.00	(0.21)	10.00	(0.15)
	10,000	10.00	(0.22)	10.00	(0.16)	10.00	(0.13)	10.00	(0.12)	10.00	(0.10)
$\hat{\tau}^2$	900	**17.24**	(9.09)	**17.24**	(9.09)	**17.24**	(9.09)	**17.24**	(9.09)	3.64	(2.89)
	1,000	**17.07**	(8.88)	3.42	(2.71)	**16.18**	(7.92)	3.42	(2.71)	3.40	(2.66)
	5,000	10.47	(3.22)	3.10	(2.10)	7.74	(1.72)	2.92	(1.80)	1.21	(0.68)
	10,000	5.47	(0.87)	2.50	(1.25)	3 .67	(0.41)	2.16	(0.87)	1.04	(0.47)
$\hat{\sigma}^2$	900	96.47	(21.03)	96.47	(21.03)	96.47	(21.03)	96.47	(21.03)	97.61	(5.72)
	1,000	96.47	(20.26)	97.77	(5.33)	95.17	(17.60)	97.77	(5.33)	97.91	(5.31)
	5,000	96.61	(3.42)	98.64	(2.36)	96.45	(2.32)	98.63	(2.31)	99.85	(2.16)
	10,000	98.08	(1.48)	99.16	(1.54)	98.42	(1.45)	99.19	(1.50)	100.00	(1.48)
$\bar{e^2}$	900	12.12	(17.52)	12.12	(17.52)	12.12	(17.52)	12.12	(17.52)	1.30	(0.46)
	1,000	11.73	(16.96)	1.27	(0.41)	9.42	(14.05)	1.27	(0.41)	1.26	(0.40)
	5,000	4.16	(2.62)	1.28	(0.41)	2.45	(0.69)	1.22	(0.31)	0.99	(0.06)
	10,000	1.92	(0.37)	1.14	(0.24)	1.33	(0.17)	1.05	(0.14)	0.94	(0.05)

Finally, Figure 4.1 presents the difference in cumulative computation time when the algorithms have to produce up-to-date parameter estimates each time a new data point arrives (or after the indicated number of data points). We scale the time required to update the estimates of the model parameters proportional to the time required to estimate the model when $n = 500$. There is no visible difference between the different variants of SEMA, which all grow linearly by factor of about 10 (as n grows with a factor of 20). Figure 4.1 shows four variants of the EM algorithm. The estimates of the model parameters are updated using EM every: 1, 10, 100, or 1,000 data points. All four variants of the EM algorithm grow with a much larger factor than SEMA when it has to produce up-to-date parameter estimates when data enter over time. More importantly the curves of the EM algorithm tend to deviate from linear and curve more upwards as larger datasets are analyzed. These curved lines of the EM algorithm illustrate that analyzing nested data using the EM algorithm when data points enter over time becomes infeasible, as the estimation of the model parameters will require increasingly more time.

To conclude, both the model-parameter estimates and the prediction errors can be improved by using better starting values obtained from a training dataset. Also, performing a single EM iteration after every 1,000 data points improved parameter estimates and lowered prediction errors. Experimentation with variants of the latter method showed that even larger improvements can be obtained by either performing multiple EM iterations, or performing the single EM iteration more frequently. In other words, depending on whether this is feasible in the streaming data application concerned, other combinations of SEMA and EM could be used.

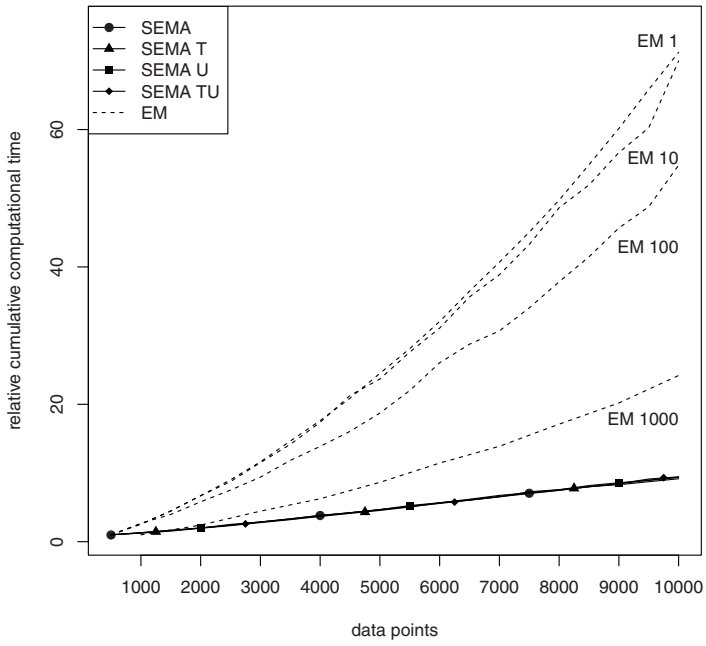


Figure 4.1: Relative cumulative computation time as a function of increasing number of data points, when the estimates of the model parameters are updated after each data point (SEMA, EM 1, or after the indicated number of data points (10, 100, or 1,000).

Table 4.7: Longitudinal happiness ratings: model parameter estimates and average squared error

	$\hat{\mu}$		$\hat{\tau}^2$		$\hat{\sigma}^2$		$\bar{e^2}$	
n	SEMA	EM	SEMA	EM	SEMA	EM	SEMA	EM
100	68.15	67.01	125.60	161.52	235.87	229.08	31.40	27.15
1,000	65.60	65.49	129.24	121.72	349.30	353.06	24.64	22.82
5,000	63.96	63.80	103.33	93.91	336.33	337.67	19.58	19.06
10,000	64.34	64.39	103.61	97.16	357.36	359.90	14.14	13.68
17,742	64.72	64.85	100.22	93.58	365.28	367.16	0.30	–

4.5 An application of SEMA to longitudinal happiness ratings

To illustrate the use of SEMA in a real-life application, we use data from a longitudinal study of happiness ratings by Killingsworth and Gilbert (2010). We will fit a random-intercept model to the data to obtain individual-level estimates of respondents' happiness. Data were collected using a smart-phone application, yielding daily measurements of the participants' happiness on a continuous rating scale. The dataset contains a total of $n = 17{,}742$ observations for $J = 2{,}248$ individuals. The average number of observation per person is 7.89, with a minimum of one observation (254 individuals) and maximum of 39 observations (one individual). While the authors analyzed the dataset after the data collection stopped, in reality, the data entered as a data stream. We fit a random-intercept model on the data stream resulting from the smart-phone application, by replaying the data collection over time.

During the data stream, we obtained parameter estimates from the SEMA and EM algorithms from only the data seen so far, and compared these to the EM estimates using the entire dataset: $\hat{\mu} = 64.85$, $\hat{\tau}^2 = 93.58$, and $\hat{\sigma}^2 = 367.16$. We used the individual-level effects estimated using all data (i.e., end of the data stream) as the "true" individual-level effect, to compute the average squared prediction error during the data stream. From these "true" estimates, we find that the estimated reliabilities range from 0.20 to 0.91 with an average of 0.67. As in the simulation study, during the data stream we monitored the estimates of $\hat{\mu}$, $\hat{\tau}^2$, and $\hat{\sigma}^2$, as well as the estimated individual-level effects. The starting values for SEMA were, respectively, $\mu_0 = y_{t=1}$ (the first observation), $\tau_0^2 = 1$, and $\sigma_0^2 = 1$.

Table 4.7 reports the values of the parameter estimates and the average squared prediction error for both SEMA and EM. Similar to the results obtained in the simulation study, $\hat{\mu}$ is estimated properly, while $\hat{\tau}^2$ is somewhat overestimated by SEMA compared to EM. The residual variance, $\hat{\sigma}^2$, is somewhat underestimated. The average squared prediction error of SEMA is close to the average squared prediction error of the EM algorithm, even though at the end of the data stream EM is obviously favored due to its own use in the operationalization of the "gold standard".

4.6 SEMA characteristics

4.6.1 Theoretical considerations

The proposed SEMA algorithm yields two improvements compared to the traditional EM algorithm. First, it is no longer required to store all n data points in memory, leading to a decrease in memory required. What needs to be stored are merely the current values of n_j, $\bar{y}_j$, $\bar{y^2_j}$, T_{1j}, T_{2j}, and T_{3j} for each of the J individuals. Second, SEMA decreases the number of computations compared to the conventional EM algorithm when analyzing a data stream. The SEMA algorithm updates the n_{j_t}, $\bar{y}_{j_t}$, $\bar{y^2_{j_t}}$, T_{1j_t}, T_{2j_t}, and T_{3j_t} for a single individual only, and subsequently updates the CDSS and the model parameters in a single pass.

SEMA is conceptually positioned between what Liang and Klein (2009) call incremental EM and stepwise EM (Cappé & Moulines, 2009). Neal and Hinton (1998) provide a proof for the large sample convergence of both stepwise and incremental EM. Incremental EM estimates the parameters by storing the CDSS and the contributions of each of the *data points* and then iterates over the dataset, subtracting previous contributions of the data point to the CDSS, thereby correcting the erroneous contribution to the CDSS of previous time. As such, incremental EM requires all data points in memory. In contrast, stepwise EM does not store all the data, but it does not correct for previous contributions to the CDSS: stepwise EM adds a weighted contribution of each data point to the CDSS. To use stepwise EM, the analyst has to choose a weight given to new data points. SEMA, which conceptually combines the two earlier methods, does not store the observed data; it stores only contributions at the level of the individuals, instead of the data points themselves. This means that SEMA scales with J, instead of n in the case of incremental EM, while using more information than stepwise EM.

4.6.2 Convergence

Fitting multilevel models on data streams adds an additional complication to standard offline methods: it is not immediately clear when (e.g., after how many observations), the parameter estimates can be said to have "converged" and thus can be substantively interpreted. However, options are available to address this issue. One could, for example choose, during the data stream, to compute a moving average of the absolute difference between two estimates of the same parameter at adjacent time points:

$$\bar{\delta}_\theta = \sum_{i=(t-C)}^{(t-1)} |\hat{\theta}_i - \hat{\theta}_{i+1}|/C, \tag{4.18}$$

where C is the size of the window of the moving average and θ is one of the model parameters. As new data points enter and thus t increases, the average will cover a new interval of parameter differences. This measure $\bar{\delta}_\theta$ – which can be maintained during the stream – can be used to quantify convergence (where given some cut-off

ζ, $\bar{\delta}_\theta < \zeta$ would imply convergence). If we examine the behavior of $\bar{\delta}_\theta$ for the simulations presented in Section 4.4, we find for the parameter μ that in all streams $\bar{\delta}_\mu$ monotonically approaches zero, and that the difference between the parameter estimates of μ obtained using our online method, and those obtained offline (where we determined convergence by no change in parameter estimates to the fourth decimal) decreases as $\bar{\delta}_\mu$ decreases. Hence, $\bar{\delta}_\theta$ seems a good candidate to use for convergence; the smaller the value of $\bar{\delta}_\theta$, the closer the parameter estimates are to their offline equivalent. The actual cut-offs ζ for $\bar{\delta}_\theta$ will be problem dependent and might differ for the different parameters. For the parameter τ^2 and σ^2, we also find that $\bar{\delta}_{\tau^2}$ and $\bar{\delta}_{\sigma^2}$ decrease during the stream, and that these decreases correspond to more and more precise estimates. However, for σ^2, the decrease is quite a bit slower (i.e., σ^2 needs more observations) than for μ, in many of the simulations, indicating that some parameters might be said to have converged sooner than others.

4.7 Extending SEMA

In practice, one might want to extend the random-intercept model to a model with more parameters. One could, for example, include covariates to improve the estimates of the model parameters and the predictions resulting from the model. Here, we discuss the inclusion of additional *fixed* effects to SEMA. This model can be written as:

$$y_{ij} = \mathbf{x}_j\beta + \mu_j + \epsilon_{ij}, \tag{4.19}$$

where $\mathbf{x}_j$ is a p-dimensional row-vector of covariates at the individual level with first element equal to 1 for the intercept, β is a p-dimensional vector with fixed-effect regression coefficients, and the individual-level intercepts μ_j are normally distributed as:

$$\mu_j \sim \mathcal{N}(0, \tau^2). \tag{4.20}$$

We assume the covariates are constant within each individual: $x_{ij} = x_j$. Because μ_j is now centered around zero, the computation of the parameters is altered slightly. In the E step the following individual-level statistics are computed:

$$\hat{\mu}_j = \hat{V}_j^{-1}(\bar{y}_j - \mathbf{x}_j\hat{\beta})n_j, \tag{4.21}$$

where $\hat{V}_j^{-1}$ equals

$$\begin{aligned} \hat{V}_j^{-1} &= \hat{\nu}_j/\hat{\sigma}^2, \\ &= \hat{\tau}^2\Big(1 - \frac{\hat{\tau}^2}{\hat{\tau}^2 + \hat{\sigma}^2/_{n_j}}\Big)\frac{1}{\hat{\sigma}^2}, \\ &= \frac{1}{\hat{\sigma}^2/_{\hat{\tau}^2} + n_j}. \end{aligned} \tag{4.22}$$

The CDSS of β, τ^2, and σ^2 are again referred to as T_1, T_2, and T_3, where T_1 is now a vector, instead of a scalar. The contributions to the CDSS for a single individual are can then be computed as follows:

$$\begin{aligned} T_{1j_t} &= n_{j_t}\mathbf{x}'_{j_t}\hat{\mu}_{j_t}, && (4.23)\\ T_{2j_t} &= \hat{\mu}^2_{j_t} + \hat{\nu}_{j_t}, && (4.24)\\ T_{3j_t} &= [(\bar{y}_{j_t} - \mathbf{x}_{j_t}\hat{\beta} - \hat{\mu}_{j_t})^2 + \hat{\nu}_{j_t}]n_{j_t}, && (4.25) \end{aligned}$$

where, as previously, $T_{w(t)} = T_{w(t-1)} - T_{wj_t(t-1)} + T_{wj_t(t)}$.

In the M step, the CDSS can be used to obtain new estimates for the model parameters using Escobar and Moser's (1993) updating method for the regression coefficients, as follows:

$$\begin{aligned} \hat{\beta} =& \Big(\sum_{j=1}^{J}\sum_{i=1}^{n_j}(\mathbf{x}'_{ij}\mathbf{x}_{ij})\Big)^{-1}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\mathbf{x}'_{ij}(y_{ij} - \hat{\mu}_j),\\ =& \Big(\sum_{j=1}^{J} n_j(\mathbf{x}'_j\mathbf{x}_j)\Big)^{-1}\sum_{j=1}^{J}\mathbf{x}'_j(\bar{y}_j - \hat{\mu}_j)n_j,\\ =& \Big(\sum_{j=1}^{J} n_j(\mathbf{x}'_j\mathbf{x}_j)\Big)^{-1}\sum_{j=1}^{J} n_j\mathbf{x}'_j\bar{y}_j - n_j\mathbf{x}'_j\hat{\mu}_j,\\ =& A_1(A_2 - T_1), \end{aligned} \tag{4.26}$$

where A_1 and A_2 can both be computed online:

$$\begin{aligned} A_{1(t)} &= A_{1(t-1)} - \frac{A_{1(t-1)}\mathbf{x}_{ij_t}\mathbf{x}'_{ij_t}A_{1(t-1)}}{1 + \mathbf{x}'_{ij_t}A_{1(t-1)}\mathbf{x}_{ij_t}}, && (4.27)\\ A_{2(t)} &= A_{2(t-1)} + \mathbf{x}'_{ij_t}y_{ij_t}. && (4.28) \end{aligned}$$

Note that, we use $\mathbf{x}_{ij_t}$ in this formulation. This means that every time a new data point arrives the values of the covariates $\mathbf{x}_{j_t}$ are retrieved from memory. Because A_1 and A_2 only consist of observed data (there are no model parameters involved) and it are sums over n observations, we do not have to correct for previous contributions.

Moreover, using the notation including the summation over n_j, the fixed effect is weighted according to the number of observations of an individual. Taking into account which individuals have more observations results in better estimates of β in the case where the individual-level effect μ_j is dependent on the number of observations of that individual, n_j. For the model introduced in Equation 4.3, if μ_j depends on n_j, $T_{1j_t} = n_j\hat{\mu}_j$ and $\hat{\mu} = T_1/n$. In our simulation studies the individual-level effects were not dependent on the number of observations. Therefore, the results using either of the two formulations will effectively be the same.

Next, the variance of the random effect, $\hat{\tau}^2$, is computed as follows:

$$\hat{\tau}^2_{(t)} = \frac{T_{2(t)}}{J}. \tag{4.29}$$

This is slightly different from the previous formulation in Equation 4.12; the difference is due to the fact that μ_j is now distributed around 0 instead of μ, since we separated the fixed effects from the random effects. Lastly, the residual variance $\hat{\sigma}^2$ is the same as it was previously,

$$\hat{\sigma}^2_{(t)} = \frac{T_{3(t)}}{n}. \tag{4.30}$$

Other interesting extensions concern the inclusion of fixed and random effects for level-1 predictors and the generalization to more than 2 levels of nesting. For those models, SEMA versions can also be formulated, which as shown here involves the derivation of the updating formulas for the expected sufficient statistics and for the parameters. In future research we will look into these extensions.

4.8 Discussion

Since data streams are becoming more common in both real-life applications and social science research (e.g., W. Hofmann, Adriaanse, Vohs, and Baumeister, 2014; Killingsworth and Gilbert 2010; Pedro, Z., Baker, Bowers, and Heffernan, 2013) there is a need for computationally feasible methods to analyze data streams. This chapter presents a novel method for estimating multilevel models in data streams consisting of dependent observations. Because the regular EM algorithm becomes computationally infeasible as the size of the data stream grows, we propose a streaming EM approximation (SEMA). SEMA is obtained by adapting the E step of the EM algorithm; that is, by using a partial E step (McLachlan & Peel, 2000) in which only the contributions to the sufficient statistics of the individual providing the new observation are updated.

Our first simulation study showed that SEMA recovers both the fixed effect and the individual-level (random) effects well, as encountered in grouped data streams. Also, the variance components are well estimated, although in conditions with very low reliability (i.e., when the residual variance is large compared to the variance of the random intercept), a large number of data points are needed to get estimates which are close to population values. In the second simulation study, we examined two ways to improve the estimates obtained by SEMA early in the data stream. First, one could occasionally preform a single EM iteration using all individuals entered so far. Using this extra information for the estimation of the model parameters resulted in parameters which approached the EM estimates of the parameters faster. Second, one could use the first n (where we choose $n = 1,000$) data points of the stream as a training set. These first data points can be used to obtain better starting values, by applying EM until convergence, after which the stream is continued using SEMA to estimate the model parameters. The combination of the two approaches showed an even larger improvement. Finally, in our implementation, an individual-level effect is updated when a new data point enters for the person concerned. However, when

individual-level prediction is the main focus of the analysis, one could fine tune the estimation of the individual-level effects, for example, by recomputing these at the moment that they are needed using the most recent model parameter estimates. The proposed alterations to SEMA, SEMA-T and SEMA-U, provide a step in this direction.

It is to be noted that the random-intercept model, as presented in Equation 4.3, which provided the basis for our SEMA algorithm, can also be formulated differently: one could also interpret the current model as a factor analysis model in which our "observations within individuals" correspond to multiple items within individuals, to which one fits a single-factor model. The current model could then be specified as: $\mu + \tau z_j + \epsilon_{ij}$, where $z_j \sim \mathcal{N}(0, 1)$. The (offline) EM algorithm to fit this model, and its generalizations, is specified in detail in Rubin and Thayer (1982). For our current model, the covariances between the items are constrained, which allows one to also in this formulation derive a online version of the EM algorithm leading to the same update steps as presented here. However, this seems not to be true in the general case: when the covariances are unconstrained, the computation of the sufficient statistics in a data stream seems cumbersome, due to the differing numbers of observations within individuals during the stream. Still, the factor-analytic view on the current problem might, in future work, inspire online EM approximations of more complex models.

Another issue to be noted is that ordering of the data points in the data stream is important for the rate of convergence of SEMA. Especially in the beginning of the data stream, if the data points are very extreme, SEMA will require more data to find the maximum-likelihood estimates for the model parameters. This is conceptually similar to using offline EM with poorly-chosen starting values of the parameters: in this case convergence will also be slow. As the data stream progresses, the influence of extreme values will lessen, since their contribution to the CDSS will decrease at a rate of at least $1/J$. Additionally, in the case that all the data for an individual enter as a block (i.e., all at once), the individual-level effect for this individual could be based on model parameters which are not yet close to the maximum of the likelihood function. This could result in contributions to the CDSS of an individual which are incorrect, and because the data entered in a block, the incorrect contributions to these CDSS are not corrected. Even though the effect of these incorrect contributions will decrease eventually, as new data points (and individuals) enter, this is an additional reason to do a full EM iteration, using all individuals, occasionally during the data stream.

With the introduction of SEMA, we provide a novel method to fit multilevel models row-by-row. This allows for the analysis of data streams and extremely large data sets, without revisiting the previous data. Because SEMA is an online method, it is not necessary to store all the data points in memory. Additionally, SEMA requires less computational power than the EM algorithm when fitting multilevel models to data streams. These two advantages make SEMA attractive both in terms of the

number of computations and in terms of the memory requirements.

Acknowledgement

We would like to thank dr. M.A. Killingsworth and prof.dr. D.T. Gilbert for sharing their data. Furthermore we would like to thank the editor and the anonymous reviewers for the great contribution to the chapter. Finally, we would like to thank James Mason, Sophia Rabe-Hesketh, and Anders Skrondal for their feedback during the writing process.

Chapter 5

Estimating Multilevel Models on Data Streams

Abstract

Social scientists are often faced with data that have a nested structure: for example, pupils are nested within schools, employees are nested within companies, or repeated measurements are nested within individuals. Data sets that have such nested structures are typically analyzed using multilevel models. However, when data sets are extremely large or when new data continuously augment the data set, estimating multilevel models can be challenging: the algorithms used to fit multilevel models repeatedly revisit all data points and end up consuming a lot of time and computer memory. This is especially troublesome when predictions are needed in real time and observations keep *streaming* in. We address this problem by introducing the Streaming Expectation-Maximization Approximation (SEMA) algorithm for fitting multilevel models online (or "row-by-row"). In a simulation study, we demonstrate the performance of SEMA compared to traditional methods of fitting multilevel models. Next, the algorithm is used to analyze an empirical data set that was originally recorded as a data stream. We show that the prediction accuracy is SEMA is both competitive and orders of magnitude faster than traditional methods.

This chapter is submitted as Ippel, L., Kaptein, M.C, & Vermunt, J.K. Estimating Multilevel Models on Data Streams

5.1 Introduction

Novel technological advances – such as the widespread use of smartphone applications – facilitate monitoring individuals over extensive periods of time (L. F. Barrett & Barrett, 2001; Beck, 2015; Buskirk & Andrus, 2012). When we monitor, for example, the behavior of customers on a webpage, patients' compliance with their medical regimen, or students' performances, we are likely interested in the behavior or traits of individuals. Based on individual-level estimates of traits, we can tailor actions or treatments; e.g., we could recommend certain books tailored to individuals' preferences as displayed by their browsing behavior. Such tailoring can only be carried out in real time when up-to-date predictions are continuously available. In this chapter, we present a computationally-efficient algorithm for generating predictions of individuals' traits in situations in which data are continuously collected.

When continuously monitoring the attitudes and behaviors of individuals, data collection is effectively never 'finished': new costumers visit websites, patients continue to see their doctors, and students enter and leave universities. This situation, in which new data enter continuously, is known as a *data stream* (Gaber, 2012; Gaber et al., 2005). Due to the continuous influx of new observations, data streams quickly result in (extremely) large data sets – possibly larger than would fit in computer memory. Even when the storage of all of these observations is technically feasible, obtaining up-to-date predictions using all available information is often computationally infeasible: the computational time to re-estimate the model parameters each time the data set is augmented often increases non-linearly and quickly becomes unacceptable. In addition, the aforementioned examples describe situations in which the collected data have a nested structure. This nesting introduces dependencies among the observations, and these dependencies in turn violate a key assumption of many statistical models that assume that observations are (conditionally) independent (Kenny & Judd, 1986). Nested structures are often dealt with using multilevel models (Goldstein & McDonald, 1988; Steenbergen & Jones, 2002) which, due to their complexity, only exaggerate the computation time problems encountered when dealing with streaming data. Since the likelihood function of a multilevel model has to be maximized iteratively (using, for example, the Expectation-Maximization algorithm, EM, Dempster et al., 1977), the computation time increases exponentially. Thus, when real-time predictions of individuals' scores are needed during a data stream efficient computational methods designed to deal with data streams, are required.

In the literature, several adaptations of the EM algorithm that are computationally more efficient than the traditional EM algorithm have been proposed. For instance, Neal and Hinton (1998) detail a number of possible adaptations to the general EM algorithm to deal with large and/or growing data sets using batches of data. These adaptations are further explained and extended in McLachlan and Peel's *Finite Mixture Models* book (2000, ch. 12) and by Thiesson et al. (2001). Wolfe et al.

(2008) discuss how to parallelize the EM algorithm to deal with extremely large data sets; a method that is less well-suited for dealing with streaming data. Finally, for a number of specific statistical models, computationally efficient versions of the EM algorithm have recently been proposed (Cappé, 2011a; Cappé & Moulines, 2009; Ippel et al., 2016b; Liu et al., 2006). The current chapter adds to this existing literature by presenting a computationally efficient algorithm for the estimation of multilevel models – or linear mixed models – in data streams.

The SEMA algorithm can be categorized as an *online*-learning algorithm. Online learning refers to "computing estimates of model parameters on-the-fly, without storing the data and by continuously updating the estimates as more observations become available" (Cappé, 2011a). A simple illustration of online learning can be provided by carefully inspecting the computation of a sample mean. The standard, *offline*, computation of a sample mean using,

$$\frac{1}{n}\sum_{t=1}^{n} x_t,$$

is inefficient since when a new data point enters, we increment n by one, and we redo our computation by revisiting all our stored data points. As a result, all data have to be available in computer memory, and the computation time grows each time a new observation is added. An *online* computation of a sample mean solves these issues. When computing the sample mean online, it is only necessary to store the sufficient statistics, n and $\bar{x}$, and these are efficiently *updated* when a new data point enters[2]:

$$\begin{aligned} n &:= n + 1 \\ \bar{x} &:= \bar{x} + \frac{x_t - \bar{x}}{n}. \end{aligned} \tag{5.1}$$

Here, n is total number of observations, $\bar{x}$ is the sample mean, and ':=' is the assignment operator, indicating that the left hand side is replaced by what is on the right-hand side. Note that we will use this operator throughout the chapter.

In this chapter, we present a fully online method for estimating multilevel models by extending the online EM algorithm introduced previously by Ippel et al. (2016b). Their so-called SEMA algorithm (Streaming Expectation Maximization Approximation) dealt only with random-intercept models. The aim of this chapter is to extend this method to allow for fitting multilevel models that contain level-1 and level-2 fixed effects, as well as random intercepts and slopes. Hence, we extend the previous work to a much broader class of linear mixed models. Throughout this chapter, we will use the terminology of repeated observations nested or grouped within individuals. However, multilevel models and the SEMA algorithm are not restricted to this type of grouping.

[2]See, for an online-estimation tutorial, Ippel et al. (2016a).

In the next section, the offline estimation of multilevel models using the EM algorithm is explained in detail. Subsequently, we illustrate the online fitting procedure of multilevel models using the SEMA algorithm. Section 5.4 presents a simulation study examining the performance of SEMA in terms of estimation accuracy and prediction error. This section is followed by an empirical example of a data stream consisting of repeated measurements. Finally, the results of both evaluations are discussed and directions for future research are highlighted.

5.2 Offline estimation of multilevel models

Here, we discuss the estimation of multilevel models using the Expectation Maximization (EM, Dempster et al., 1977) algorithm. Multilevel models can contain both fixed effects and random effects. Fixed effects, which we denote using $\boldsymbol{\beta}$, are assumed to have the same effect across individuals. Effects which are assumed to vary between individuals are random effects ($\boldsymbol{b}_j$). For example, including a random intercept formalizes the assumption that individuals have different starting points, though the covariates still affect all individuals equally. A random slope effectively adds a distribution of effects of a covariate, such that a covariate can affect individuals differently.

Let individual j have $i = 1, \ldots, n_j$ observations and let $n = \sum_{j=1}^{J} n_j$ be total number of observations collected from J individuals. The model fitted to the data is:

$$y_{ij} = \boldsymbol{x}'_{ij}\boldsymbol{\beta} + \boldsymbol{z}'_{ij}\boldsymbol{b}_j + \epsilon_{ij}, \tag{5.2}$$

$$\boldsymbol{b}_j \sim \mathcal{MVN}(0, \boldsymbol{\Phi})$$

$$\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$$

where

- y_{ij} is the response i of individual j,
- $\boldsymbol{x}_{ij}$ is a $p \times 1$ vector of *fixed* effect data,
- $\boldsymbol{z}_{ij}$ is a $r \times 1$ vector of *random* effects data,
- $\boldsymbol{\beta}$ is a $p \times 1$ vector of fixed-effect coefficients,
- $\boldsymbol{b}_j$ is a $r \times 1$ vector of random effects coefficients,
- $\boldsymbol{\Phi}$ is a $r \times r$ matrix with (co)variances of the random effects,
- ϵ_{ij} is the error term for each observation,
- σ^2 is the variance of the error term

The number of observations per individual, n_j, might differ across individuals. Furthermore, the variance of the random effects and the error variance are assumed to be independent: $\boldsymbol{\epsilon} \perp \boldsymbol{b}_j$.

Often, the maximum likelihood framework is used to estimate the model parameters of the above multilevel model. If the random effects ($\boldsymbol{b}_j$) would have been observed, optimizing the log-likelihood function would be straightforward. The log-likelihood function has the following distribution:

$$\begin{aligned}\ell(\boldsymbol{\beta}, \boldsymbol{\Phi}, \sigma^2 | y, \boldsymbol{b}_j) = -\frac{n}{2}\ln\sigma^2 - \frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\Big(\frac{(y_{ij} - \boldsymbol{x}'_{ij}\boldsymbol{\beta} - \boldsymbol{z}'_{ij}\boldsymbol{b}_j)}{\sigma}\Big)^2 \\ -\frac{J}{2}\ln|\boldsymbol{\Phi}| - \frac{1}{2}\sum_{j=1}^{J}\boldsymbol{b}'_j\boldsymbol{\Phi}^{-1}\boldsymbol{b}_j\end{aligned} \tag{5.3}$$

However, since these random effects are not directly observed (i.e., these are latent) we are confronted with a missing-data problem. One approach to deal with this missing-data problem is using the EM algorithm to maximize the log-likelihood function. By imputing these missing values with the expectations of $\boldsymbol{b}_j$'s given the model parameters $\boldsymbol{\beta}$, $\boldsymbol{\Phi}$, and σ^2 in the E-step and subsequently maximizing the log-likelihood function given these expectations in the M-step, the EM algorithm iteratively finds the parameter values that maximize the likelihood. Below, first the details regarding the computation in the E-step are presented, after which the computations of the M-step are presented.

5.2.1 The offline E-step

When the missing values, $\boldsymbol{b}_j$'s, are replaced by their expected values given the current parameter estimates and the available data of individual j there are closed-form expressions to compute the model parameters. These closed-form expressions are based on a number of *complete-data sufficient statistics* (CDSS), which are computed as part of the E-step. Each of the model parameters has its own CDSS. We refer to the three necessary CDSS as $\boldsymbol{t}_1$, $\boldsymbol{T}_2$, and t_3 (for respectively $\boldsymbol{\beta}$, $\boldsymbol{\Phi}$, and σ^2).

The CDSS for $\boldsymbol{\beta}$ is defined as follows:

$$\boldsymbol{t}_{1(k)} = \sum_{j=1}^{J}\boldsymbol{X}'_j\boldsymbol{Z}_j\hat{\boldsymbol{b}}_{j(k)}, \tag{5.4}$$

where $\boldsymbol{X}_j$ is an $n_j \times p$ matrix, $\boldsymbol{Z}_j$ is $n_j \times r$ matrix, k indexes the current iteration, $\boldsymbol{t}_{1(k)}$ is an $p \times 1$ vector, and $\hat{\boldsymbol{b}}_{j(k)}$ is given by:

$$\begin{aligned}\hat{\boldsymbol{b}}_{j(k)} =& \boldsymbol{C}^{-1}_{j(k)}\boldsymbol{Z}'_j(\boldsymbol{y}_j - \boldsymbol{X}_j\hat{\boldsymbol{\beta}}_{(k-1)}), \\ =& \boldsymbol{C}^{-1}_{j(k)}(\boldsymbol{Z}'_j\boldsymbol{y}_j - \boldsymbol{Z}'_j\boldsymbol{X}_j\hat{\boldsymbol{\beta}}_{(k-1)}).\end{aligned} \tag{5.5}$$

Here $\boldsymbol{C}_{j(k)}$ quantifies the uncertainty of the imputations of $\boldsymbol{b}_j$'s, and $k-1$ indicates that $\hat{\boldsymbol{\beta}}$ of the previous iteration is used in the computation.[3] $\boldsymbol{C}_{j(k)}$ itself is an $r \times r$

[3] For more details and proof, see Raudenbush and Bryk (2002), Ch. 14

matrix:

$$\boldsymbol{C}_{j(k)} = \boldsymbol{Z}_j'\boldsymbol{Z}_j + \hat{\sigma}^2_{(k-1)}\hat{\boldsymbol{\Phi}}^{-1}_{(k-1)}. \tag{5.6}$$

The estimation of the complete-data sufficient statistic for the variance of the random effect, $\boldsymbol{T}_{2(k)}$, is given by:

$$\boldsymbol{T}_{2(k)} = \sum_{j=1}^{J} \hat{\boldsymbol{b}}_{j(k)}\hat{\boldsymbol{b}}'_{j(k)} + \hat{\sigma}^2_{(k-1)} \sum_{j=1}^{J} \boldsymbol{C}^{-1}_{j(k)}, \tag{5.7}$$

where $\boldsymbol{T}_{2(k)}$ is an $r \times r$ matrix. In words, $\boldsymbol{T}_{2(k)}$ is the sum of the squared random-effects coefficients plus the additional uncertainty due to the fact that $\boldsymbol{b}_{j(k)}$ is not observed.

Lastly, the complete-data sufficient statistic of the residual variance, $\sigma^2_{(k)}$, $t_{3(k)}$ is given by:

$$t_{3(k)} = \sum_{j=1}^{J} u'u + \hat{\sigma}^2_{(k-1)} tr \sum_{j=1}^{J} \boldsymbol{C}^{-1}_{j(k)}\boldsymbol{Z}_j'\boldsymbol{Z}_j. \tag{5.8}$$

where $u = \boldsymbol{y}_j - \boldsymbol{X}_j\hat{\boldsymbol{\beta}}_{(k-1)} - \boldsymbol{Z}_j\hat{\boldsymbol{b}}_{j(k)}$, is the standard residual.

5.2.2 The offline M-step

In the M-step, the log-likelihood function is maximized, given the CDSS of E-step. While presenting the computations, we also indicate which parts present difficulties when operating on a data stream. We discuss the computation of each of the model parameters in turn, starting with the fixed effects ($\boldsymbol{\beta}$).

In iteration k, the coefficients of the fixed effects are computed using the normal equations:

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{(k)} =& \Big(\sum_{j=1}^{J} \boldsymbol{X}_j'\boldsymbol{X}_j\Big)^{-1} \sum_{j=1}^{J} \boldsymbol{X}_j'(\boldsymbol{y}_j - \boldsymbol{Z}_j\hat{\boldsymbol{b}}_{j(k)}),\\
=& \Big(\sum_{j=1}^{J} \boldsymbol{X}_j'\boldsymbol{X}_j\Big)^{-1} \sum_{j=1}^{J} \boldsymbol{X}_j'\boldsymbol{y}_j - \boldsymbol{X}_j'\boldsymbol{Z}_j\hat{\boldsymbol{b}}_{j(k)},\\
=& \Big(\sum_{j=1}^{J} \boldsymbol{X}_j'\boldsymbol{X}_j\Big)^{-1} \sum_{j=1}^{J} \boldsymbol{X}_j'\boldsymbol{y}_j - \boldsymbol{t}_{1(k)}.
\end{aligned} \tag{5.9}$$

Equation 5.9 has multiple elements which are difficult to compute in a data stream. For instance, the matrix multiplication and summation of the matrices for each individual ($\sum_{j=1}^{J} \boldsymbol{X}_j'\boldsymbol{X}_j$), and the resulting $p \times p$ matrix inversion are computationally expensive when there are many covariates.

The variance of the random effects ($\boldsymbol{\Phi}_{(k)}$) is computed by dividing $\boldsymbol{T}_{2(k)}$ by the number of individuals:

$$\begin{aligned}\hat{\boldsymbol{\Phi}}_{(k)} &= \frac{\sum_{j=1}^{J} \hat{\boldsymbol{b}}_{j(k)}\hat{\boldsymbol{b}}'_{j(k)} + \hat{\sigma}^2_{(k-1)} \sum_{j=1}^{J} \boldsymbol{C}^{-1}_{j(k)}}{J}, \\ &= \frac{\boldsymbol{T}_{2(k)}}{J}.\end{aligned} \tag{5.10}$$

Lastly, the residual variance ($\sigma^2_{(k)}$) is computed as follows:

$$\begin{aligned}\hat{\sigma}^2_{(k)} &= \frac{\sum_{j=1}^{J} u'u + \hat{\sigma}^2_{(k-1)} tr \sum_{j=1}^{J} \boldsymbol{C}^{-1}_{j(k)} \boldsymbol{Z}'_j \boldsymbol{Z}_j}{n}. \\ &= \frac{t_{3(k)}}{n}\end{aligned} \tag{5.11}$$

The latter equation again illustrates that fitting multilevel models in a data stream is computationally intensive. The residual variance is computed by estimating the residual for each observation. Using this offline formulation of σ^2, all observations thus have to be stored in memory. Furthermore, the residual depends on the model parameters of the previous iteration. Because the model parameters change with each iteration, the residual, $y_{ij} - \boldsymbol{x}'_{ij}\hat{\boldsymbol{\beta}}_{(k)} - \boldsymbol{z}'_{ij}\hat{\boldsymbol{b}}_{j(k)}$, changes accordingly and needs to be re-computed.

5.3 Online estimation of multilevel models

In this section, we introduce the Streaming Expectation Maximization Approximation (SEMA) algorithm. The approximation of the E-step is presented first, followed by the M-step. At the end of this section, the full algorithm (see Algorithm 1) is described. This latter overview illustrates the sequence of computations and details which elements are stored in memory.

5.3.1 The online E-step

Previously, we used subscript k to indicate the iteration cycles of the EM algorithm. In this section, we drop this subscript to emphasize that unlike the EM algorithm, the SEMA algorithm only updates the CDSS using a single data point, without revisiting previous data points. Note that, data point refers to a vector with an identifier for an individual, the covariates with fixed effects and random effects and the observation of the dependent variable. When a data point enters, the SEMA algorithm performs an E-step only for the individual that belongs to the data point that recently entered. After the E-step for this individual, all three model parameters are updated in the M-step. Because of this updating scheme, SEMA updates the parameter estimates when a new data point enters, instead of fitting the multilevel model all over again. First, the online implementation of the CDSS for $\hat{\boldsymbol{\beta}}$ is presented.

As highlighted above, two aspects of Eq. 5.4: $\boldsymbol{t}_1 = \sum_{j=1}^{J} \boldsymbol{X}_j' \boldsymbol{Z}_j \hat{\boldsymbol{b}}_j$ are challenging in the context of a data stream. First, the CDSS for $\hat{\boldsymbol{\beta}}$ consists of a summation over J individuals. If the (weighted) contribution of a new data point would simply be added, then this would result in counting the same individual repeatedly. Second, the equation of $\boldsymbol{t}_1$ uses $\hat{\boldsymbol{b}}_j$, which depends on the model parameters. Because these model parameters are updated when a new data point enters, obtaining the exact same result using either the online or offline computation of this CDSS, would imply that all contributions to $\boldsymbol{t}_1$ need to be recomputed when new data enter. The latter, however, is not feasible, especially when the number of individuals is large. Therefore, we resort to an approximate solution. Note that this approximation becomes increasingly precise as the number of observations per individual grows.

The solution we chose is as follows: when a new data point enters, the contribution of the individual belonging to this data point is subtracted from $\boldsymbol{t}_1$ to account for the fact that this individual has already contributed to $\boldsymbol{t}_1$. Next, $\hat{\boldsymbol{b}}_j$ of this individual is recomputed, such that the new contribution to $\boldsymbol{t}_1$ of this individual can be added. Because the online implementation of the CDSS is not exactly the same as the offline CDSS, we refer to the online implemented complete-data sufficient statistic of the coefficients of the fixed effects as $\tilde{\boldsymbol{t}}_1$. The contribution to $\tilde{\boldsymbol{t}}_1$ a single individual is referred to as $\boldsymbol{t}_{1_j}$. So, defining the CDSS for $\hat{\boldsymbol{\beta}}$ as

$$\tilde{\boldsymbol{t}}_{1(t)} := \tilde{\boldsymbol{t}}_{1(t-1)} - \boldsymbol{t}_{1_{j_{t(t-1)}}} + \boldsymbol{t}_{1_{j_{t(t)}}}, \tag{5.12}$$

where $\boldsymbol{t}_{1_{j_{t(t-1)}}}$ represents the previous contribution of individual j_t, which is the individual belonging to the most recent data point, to $\tilde{\boldsymbol{t}}_1$ and $\boldsymbol{t}_{1_{j_{t(t)}}}$ is the current contribution to $\tilde{\boldsymbol{t}}_1$. Only for the CDSS, we use subscript t to indicate that the CDSS is obtained by subtracting the previous contribution of individual j_t after which the new contribution is added. The computation of $\boldsymbol{t}_{1j}$ is given by

$$\boldsymbol{t}_{1j} = \boldsymbol{X}_j' \boldsymbol{Z}_j \hat{\boldsymbol{b}}_j, \tag{5.13}$$

where the $\boldsymbol{X}_j' \boldsymbol{Z}_j$ matrix can be update online:

$$\boldsymbol{X}_j' \boldsymbol{Z}_j := \boldsymbol{X}_j' \boldsymbol{Z}_j + \boldsymbol{x}_{ij} \boldsymbol{z}_{ij}', \tag{5.14}$$

where $\boldsymbol{X}_j' \boldsymbol{Z}_j$ is the result of the matrix multiplication which is only updated for the individual belonging to the most recent data point, and $\boldsymbol{x}_{ij}$ and $\boldsymbol{z}_{ij}'$ are the new values of fixed effects and random effects covariates of this individual. Unlike Eq. 5.12, Eq. 5.14 is exact. Using Eq. 5.14, none of the data points themselves ($\boldsymbol{x}_{ij}$ and $\boldsymbol{z}_{ij}$) need to be stored since only the results of the matrix multiplication is stored. When new data present themselves, the outer product of $\boldsymbol{x}_{ij} \boldsymbol{z}_{ij}'$ is merely added to the current result of the matrix multiplication.

Next, the coefficients of the random effects (Eq. 5.5: $\hat{\boldsymbol{b}}_j = \boldsymbol{C}_j^{-1}(\boldsymbol{Z}_j' \boldsymbol{y}_j - \boldsymbol{Z}_j' \boldsymbol{X}_j \hat{\boldsymbol{\beta}})$) can similarly be approximated online. The computation of $\hat{\boldsymbol{b}}_j$ is computationally

complex due to the inversion of two matrices each time the model parameters are updated. However, when the number of random effects is small, the matrix inversion is computationally not too expensive. We first explain how $\boldsymbol{C}_j$ (Eq. 5.6) is computed online. The computation of $\boldsymbol{C}_j$ uses a matrix product of the data used for the estimation of the random effects. We define the result of the matrix multiplication $\boldsymbol{Z}_j'\boldsymbol{Z}_j$ as $\boldsymbol{Z}_j^2$. When new data enter $\boldsymbol{Z}_j^2$ can be updated as follows:

$$\boldsymbol{Z}_j^2 := \boldsymbol{Z}_j^2 + \boldsymbol{z}_{ij}\boldsymbol{z}_{ij}', \tag{5.15}$$

which is similar to Eq. 5.14. The $\boldsymbol{Z}_j^2$ matrix needs to be stored per individual. The online computation of $\boldsymbol{C}_j$ is given by:

$$\boldsymbol{C}_j = \boldsymbol{Z}_j^2 + \hat{\sigma}^2\hat{\boldsymbol{\Phi}}^{-1}. \tag{5.16}$$

Using the online formulation of $\boldsymbol{C}_j$, the next step needed for computing $\hat{\boldsymbol{b}}_j$ is given by:

$$\boldsymbol{z}_j y_j := \boldsymbol{z}_j y_j + \boldsymbol{z}_{ij} y_{ij}, \tag{5.17}$$

where $\boldsymbol{z}_j y_j$ is an $r \times 1$ vector. Note that the matrix multiplication $\boldsymbol{Z}_j'\boldsymbol{X}_j$ (see, Eq. 5.5) is equal to the transpose of the matrix $\boldsymbol{X}_j'\boldsymbol{Z}_j$ in Eq. 5.14. The online computation of $\hat{\boldsymbol{b}}_j$ is thus given by:

$$\hat{\boldsymbol{b}}_j = \boldsymbol{C}_j^{-1}(\boldsymbol{z}_j y_j - (\boldsymbol{X}'\boldsymbol{Z}_j)'\hat{\boldsymbol{\beta}}) \tag{5.18}$$

Next, we present the online computation of the CDSS for the variance of the latent variables. Similar to the computation of $\tilde{\boldsymbol{t}}_1$, $\tilde{\boldsymbol{T}}_2$ is also a summation over individuals (Eq. 5.7: $\boldsymbol{T}_2 = \sum_{j=1}^{J}\hat{\boldsymbol{b}}_j\hat{\boldsymbol{b}}_j' + \hat{\sigma}^2\sum_{j=1}^{J}\boldsymbol{C}_j^{-1}$). Therefore, a similar update regime is used for this CDSS:

$$\tilde{\boldsymbol{T}}_{2(t)} := \tilde{\boldsymbol{T}}_{2(t-1)} - \boldsymbol{T}_{2j_t(t-1)} + \boldsymbol{T}_{2j_t(t)}, \tag{5.19}$$

where

$$\boldsymbol{T}_{2j} = \hat{\boldsymbol{b}}_j\hat{\boldsymbol{b}}_j' + \hat{\sigma}^2\sum_{j=1}^{J}\boldsymbol{C}_j^{-1}. \tag{5.20}$$

Thus, in order to update $\tilde{\boldsymbol{T}}_2$ online, the previous contribution of this individual is again subtracted before the new contribution is computed and added.

Lastly, we illustrate the online computation of the CDSS for the residual variance (Eq. 5.8: $t_3 = \sum_{j=1}^{J}\sum_{i=1}^{n_j}(y_{ij} - \boldsymbol{x}_{ij}'\hat{\boldsymbol{\beta}} - \boldsymbol{z}_{ij}'\hat{\boldsymbol{b}}_j)^2 + \sigma^2 tr\sum_{j=1}^{J}\sum_{i=1}^{n_j}\boldsymbol{C}_j^{-1}\boldsymbol{z}_{ij}\boldsymbol{z}_{ij}'$). The computation of t_3 is unlike the previous two CDSS, a summation over n data points. Therefore, we first rewrite the contribution of each single data point, as a contribution of an individual to the $\tilde{t}_3$:

$$\begin{aligned} t_{3j} = y_j^2 + \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}'\boldsymbol{X}_j^2 + \hat{\boldsymbol{b}}_j\hat{\boldsymbol{b}}_j'\boldsymbol{Z}_j^2 - 2\hat{\boldsymbol{\beta}}'\boldsymbol{x}_j y_j - 2\hat{\boldsymbol{b}}_j'\boldsymbol{z}_j y_j \\ + 2\boldsymbol{X}_j'\boldsymbol{Z}_j\hat{\boldsymbol{\beta}}\hat{\boldsymbol{b}}_j' + \hat{\sigma}^2 tr\boldsymbol{C}_j^{-1}, \end{aligned} \tag{5.21}$$

where y_j^2 is computed as the sum of the squared observations of the dependent variable: $\sum_{i=1}^{n_j} y_{ij}^2$. Note that the computation of $\boldsymbol{X}_j^2$ is similar to that of $\boldsymbol{Z}_j^2$ and is described in detail below in the M-step. Using Eq. 5.21, $\tilde{t}_3$ can be updated similarly to the other CDSS:

$$\tilde{t}_{3(t)} := \tilde{t}_{3(t-1)} - t_{3j_t(t-1)} + t_{3j_t(t)}, \tag{5.22}$$

The online implementation of the E-step makes it possible to ignore the historical data points and only store summaries of the data points (see for exact details Algorithm 1 below). Next, the online implementation of the M-step is presented.

5.3.2 The online M-step

The online implementation of the M-step of both the variance of the random effects, $\hat{\boldsymbol{\Phi}} = \frac{\tilde{\boldsymbol{T}}_2}{J}$, and the residual variance, $\hat{\sigma}^2 = \frac{\tilde{t}_3}{n}$, is the same as the offline implementation of the M-step we discussed above. This, however, does not hold for the online computation of $\hat{\boldsymbol{\beta}} = \left(\sum_{j=1}^{J} \boldsymbol{X}_j' \boldsymbol{X}_j\right)^{-1} \sum_{j=1}^{J} \boldsymbol{X}_j' \boldsymbol{y}_j - \tilde{\boldsymbol{t}}_1$, which we detail in this section.

The first element of Eq. 5.9 is the $\sum_{j=1}^{J} \boldsymbol{X}_j' \boldsymbol{X}_j$ matrix. This matrix can be updated online using the same update regime as already discussed in Eq. 5.14:

$$\boldsymbol{X}^2 := \boldsymbol{X}^2 + \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}', \tag{5.23}$$

where, similar to Eq. 5.15, $\boldsymbol{X}'\boldsymbol{X}$ is defined as $\boldsymbol{X}^2$. However, in order to subsequently compute $\hat{\boldsymbol{\beta}}$, the inverse of $\boldsymbol{X}^2$ is needed. Computing the inverse of a matrix can be a costly procedure if the number of covariates is large. A solution is to directly update the inverted matrix using the Sherman–Morrison formula (Escobar & Moser, 1993; Plackett, 1950; Sherman & Morrison, 1950):

$$\boldsymbol{X}_{inv}^2 := \boldsymbol{X}_{inv}^2 - \frac{\boldsymbol{X}_{inv}^2 \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}' \boldsymbol{X}_{inv}^2}{1 + \boldsymbol{x}_{ij}' \boldsymbol{X}_{inv}^2 \boldsymbol{x}_{ij}}. \tag{5.24}$$

Using this formulation, $\boldsymbol{X}^2$ only has to be inverted once, after which the inverted matrix is directly updated with the new data. In practice, this means that one has to wait until enough data have entered, such that $\boldsymbol{X}^2$ is invertible. Directly updating the inverted matrix $\boldsymbol{X}_{inv}^2$ is more efficient than the offline estimation procedure, because the offline estimation procedure stores all observations in memory and has to invert the $\boldsymbol{X}'\boldsymbol{X}$ matrix every time new data present themselves in order to obtain up-to-date model parameters. The second part of Eq. 5.9 is the multiplication of the covariates with the dependent variable. This can be updated online as follows:

$$\boldsymbol{x}y := \boldsymbol{x}y + \boldsymbol{x}_{ij} y_{ij}, \tag{5.25}$$

where $\boldsymbol{x}y$ is a $p \times 1$ vector. Inserting the online computed components of Equation 5.9 into the equation results in the computation of $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}_{inv}^2 (\boldsymbol{x}y - \tilde{\boldsymbol{t}}_1) \tag{5.26}$$

We present a schematic overview of the SEMA algorithm, assuming that $\boldsymbol{X}^2$ is already inverted, in Algorithm 1. The first line indicates which elements the algorithm uses, where θ are all elements which should be available at the global level, whereas θ_j contains all the elements which should be stored for each individual. Only θ_j for the individual that belongs to the most recent data point is used in the update step, the remaining θ_j's do not have to be available while updating the global parameters or the elements of the individual belonging to the recently entered data point. The SEMA function in [R](R Core Team, 2016) can be found at `http://github.com/L-Ippel/SEMA_extended`.

Algorithm 1 SEMA: Notation and equations can be found in the second and third section of this chapter.

input: $\boldsymbol{x}_{ij}$, $\boldsymbol{z}_{ij}$, y_{ij}, θ, θ_j
$\theta = n, J, \boldsymbol{J}, \boldsymbol{X}^2_{inv}, \boldsymbol{x}y, \hat{\boldsymbol{\beta}}, \tilde{\boldsymbol{t}}_1, \hat{\boldsymbol{\Phi}}, \tilde{\boldsymbol{T}}_2, \hat{\sigma}^2, \tilde{t}_3$
$\theta_j = n_j, y_j^2, \boldsymbol{b}_j, \boldsymbol{Z}_j^2, \boldsymbol{X}_j'\boldsymbol{Z}_j, \boldsymbol{C}_j, \boldsymbol{X}_j^2, \boldsymbol{x}_j y_j, \boldsymbol{z}_j y_j, \boldsymbol{t}_{1j}, \boldsymbol{T}_{2j}, t_{3j}$
for t in data stream **do**
  if j_t is unknown **then**
    $\boldsymbol{J} \leftarrow \{\boldsymbol{J}, j_t\}$ ▷ $\boldsymbol{J}$ is vector with identifiers
    $J \leftarrow J + 1$ ▷ J is the length of vector $\boldsymbol{J}$
    create new record for j_t
  end if
  ▷ update global parameters
  $n \leftarrow n + 1$
  $\boldsymbol{x}y$, $\boldsymbol{X}^2_{inv}$ (Eq. 5.25 and 5.24)
  ▷ update individual parameters
  $n_j \leftarrow n_j + 1$
  $y_j^2 \leftarrow y_j^2 + y_{ij}^2$
  $\boldsymbol{X}_j^2$ (Eq. 5.23), $\boldsymbol{x}_j y_j$ (Eq. 5.25), $\boldsymbol{Z}_j^2$ (Eq. 5.15), $\boldsymbol{X}_j'\boldsymbol{Z}_j$ (Eq. 5.14), $\boldsymbol{z}_j y_j$ (Eq. 5.17)
  ▷ E-step
  compute $\boldsymbol{C}_j, \boldsymbol{b}_j$ (Eq. 5.16 and 5.18)
  compute $\boldsymbol{t}_{1j}, \boldsymbol{T}_{2j}, t_{3j}$ (Eq. 5.13, 5.20, and 5.21,)
  update $\tilde{\boldsymbol{t}}_1, \tilde{\boldsymbol{T}}_2, \tilde{t}_3$ (Eq. 5.12, 5.19, and 5.22)
  ▷ M-step
  compute model parameters $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\Phi}}, \hat{\sigma}^2$ (Eq. 5.26, 5.10, and 5.11)
  return $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\Phi}}, \hat{\sigma}^2$
end for

5.4 Simulation study

5.4.1 Design

In this section, SEMA is compared with an often used offline procedure for fitting multilevel models using simulations. As a comparison point, we use the default optimizer of the Lmer function (Bates, Mächler, Bolker, & Walker, 2015; R Core Team, 2016), which is coined "bobyqa" (acronym for Bound Optimization BY Quadratic Approximation, Powell, 2009). This algorithm finds the best fitting parameter values

by iteratively approximating the likelihood function using quadratic approximation, i.e., this algorithm does not use first or second order derivatives. We choose this comparison specifically since it is an often-used and robust implementation of the estimation of linear mixed models. SEMA and this state-of-the-art offline procedure are compared in terms of average squared prediction error ($\bar{e^2} = \frac{1}{n}\sum(\hat{y}_{ij} - y_{ij})^2$) and parameter estimation precision. We explicitly study the effect of three factors: the number of observations per individual (n_j), the number of random effects (r), and the number of level 1 covariates (lvl_1).

The number of observations n_j is an important contributor to the reliability of the estimates of $\boldsymbol{b}_j$: more observations per individual results in less uncertainty. Therefore, we expect SEMA will learn the true parameter values in conditions with a lower number of observations more slowly (i.e., more data points have to enter) than in conditions where individuals are observed more often. For the error, we expect that when individuals are observed more often, the average squared prediction error will be lower than in the case where the individuals only have a small number of observations. For the second factor, the number of random effects (r), we expect that more random effects will result in a slower rate of finding the parameter values, i.e., SEMA has to take more steps than in the condition where the number of random effects is small. Also, when the number of random effects is high, we expect that SEMA will produce more error. Lastly, for the number of covariates on the first level (lvl_1), we have similar expectations for the number of random effects: more fixed effects will lead to a slower retrieval of the data-generating parameters, and will result in more error. The three factors are all crossed, however, we will not let the number of covariates with a random effect exceed the number of covariates; hence we have the following six conditions[4]:

- $n_j = 10, r = 2$, and $\text{lvl}_1 = 3$,
- $n_j = 50, r = 2$, and $\text{lvl}_1 = 3$,
- $n_j = 10, r = 2$, and $\text{lvl}_1 = 8$,
- $n_j = 50, r = 2$, and $\text{lvl}_1 = 8$,
- $n_j = 10, r = 7$, and $\text{lvl}_1 = 8$, and
- $n_j = 50, r = 7$, and $\text{lvl}_1 = 8$.

The data stream is generated with variance terms of the random effects equal to:

- $r = 2$: $\phi^2 = 4$ and 9, or
- $r = 7$: $\phi^2 = 4$, 9, 16, 2.25, 6.25, 12.25, and 20.25.

The fixed effects are generated with the following parameter values:

[4]The number of random effects includes the random intercept.

- Number of level 1 predictors = 3: $\beta = 3.5, -4.5$, and -5.5
- Number of level 1 predictors = 8: $\beta = 3.5, 4.5, -5.5, -2.5, -3.5, -4.5, 5.5$, and 6.5

Additionally, the residual variance, the number of level 2 variables, and the length of the data stream were fixed across the conditions to

- $\sigma^2 = 25$,
- $\beta = 1.5$ and 2.5; and
- $n = 50,000$ data points, resulting in $J = 5,000$ or $J = 1,000$.

The data stream was generated by randomly sample individuals with replacement, resulting in unequal number of observations per individual. Due to the computational complexity of the offline fitting procedure, the Lmer function is fitted to the data stream only every $n = 1,000$ data points instead of after each data point. Each condition was replicated $m = 100$ times.

5.4.2 Results

Figure 5.1 presents the average estimated variance terms of both the residual variance and the variance of the random intercept and slope over the 100 replications. On the x-axes the length of the data stream is presented, and on the y-axes the parameter estimates are presented. The three figures on the left all have $n_j = 10$, and the figures on the right have $n_j = 50$. The three rows are from top to bottom: $r = 2$, $\text{lvl}_1 = 3$; $r = 2$, $\text{lvl}_1 = 8$; and $r = 7$, $\text{lvl}_1 = 8$. The gray lines represent the parameter estimates obtained using Lmer, the black lines the parameters estimates of SEMA. The Lmer function was not always able to converge in the beginning of the data stream, in these cases the gray lines are omitted from the figure. A comparison between the $n_j = 50$ and the $n_j = 10$ conditions shows that SEMA rapidly approaches Lmer's parameter estimates, especially when the number of observations for each individual is large. Furthermore, SEMA provides estimates even when Lmer is unable to converge. There is hardly any difference between including 3 level 1 predictors or 8 level 1 predictors: the top two figures and the two figures in the middle row are, given the number of observations per individual, very similar.

The bottom two figures deviate from the figures above, because these conditions are, even for the Lmer algorithm, very difficult to fit. In the bottom left figure, Lmer is only able to fit the model when at least 34,000 data points are available. Even when Lmer is able to fit the model, Lmer cycled through the data thousands of times to obtain convergence. While Lmer is able to fit the model using less data in the lower right panel than in the lower left panel, still Lmer revisited the same data thousands of times to fit the model and hence took (very) long times to compute. Comparing the results of SEMA in the panel in the lower left panel with the results of SEMA in the other panels, it is clear that this lower left panel (condition: $n_j = 10, r = 7$,

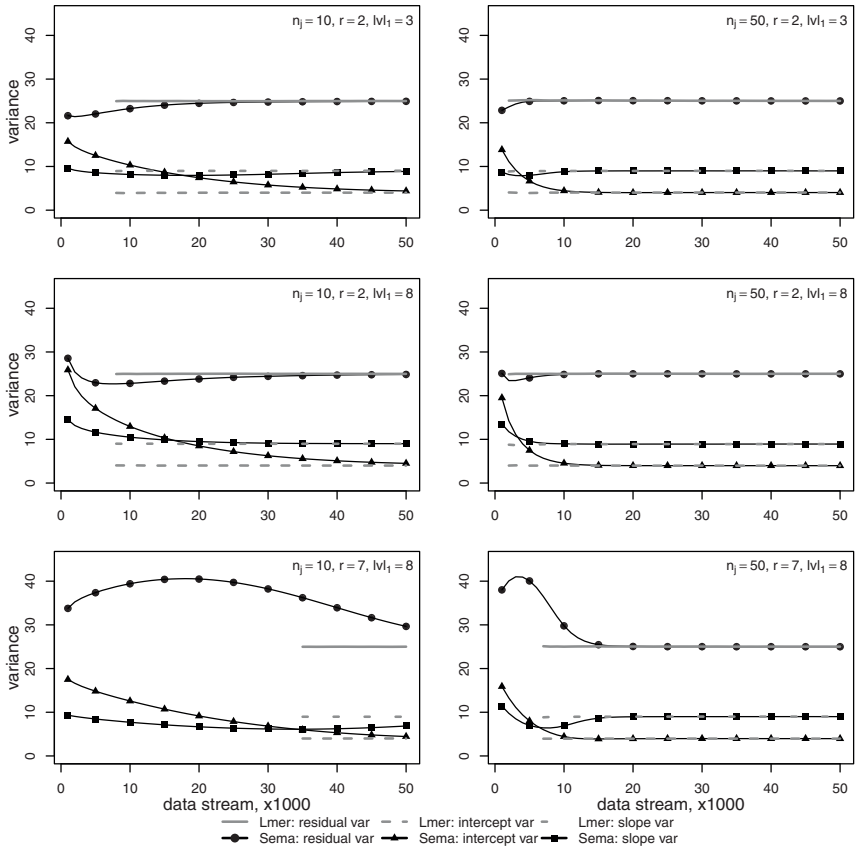


Figure 5.1: Estimated residual variance and random intercept and random slope. Note that for two graphs on the bottom, not all variance terms are included in the graph.

$\text{lvl}_1 = 8$) is, as expected, a difficult condition. Especially in the extremely challenging condition where only 10 observations per individual are available to estimate a large number of fixed and random effects, it is clear that the parameter estimates of SEMA have not yet converged, even at the end of the data stream. However, when there is more information available per individual ($n_j = 50$), SEMA performs much better (lower right panel) than in the condition with $n_j = 10$.

Next, we present three tables with the parameter estimates averaged over the replications, the standard deviation over the replications and the 95% interval based on the empirical distribution of results of the simulation study (percentiles). Table 5.1 presents the estimates of two of the fixed effects estimated by SEMA and Lmer at three points in the data stream. Since the (qualitative) behavior is similar across all fixed effects, we choose to present only these two. First note that when $n \leq 1,000$, Lmer was never able to converge, while the estimates of SEMA, even though the

empirical interval is quite wide in some conditions, are already approximating the value with which the data were generated ($\beta = 1.5$ and -5.5). Across conditions, we can conclude from Table 5.1 that the fixed effects are estimated well by both Lmer and SEMA, with the mere difference that the SD's of SEMA are slightly larger (however, SEMA is magnitudes faster).

Table 5.2 presents the estimates of the variance of the random intercept and one of the random slopes ($\phi^2 = 4$, and 9). Clearly, these variance terms are more difficult to estimate for SEMA than the fixed effects. While Lmer retrieves the values used to generate the data as soon as it is able to fit the model, SEMA uses more data to obtain good estimates of the variance terms. Notice that the average estimated value of the variance terms ($\hat{\phi}^2$) is not always positioned approximately in the middle of the empirical interval, see for instance condition $n_j = 10, r = 2$, $\text{lvl}_1 = 8$, when $n = 1,000$, both for $\phi^2 = 4$, and 9. An additional inspection of the replications lets us conclude that there were a handful of extreme outliers. When more data have entered, the effect of the outliers fades quickly: the SD becomes much smaller and the empirical interval becomes smaller. Unfortunately, we cannot compare the results of SEMA with Lmer in the runs in these extreme cases since in these cases Lmer was unable to converge. We contend that these conditions are generally very difficult.

Lastly, Table 5.3 presents the estimates of the residual variance ($\sigma^2 = 25$). The same conditions which showed a large SD for $\hat{\phi}$, have a large SD for the estimates of $\hat{\sigma}^2$. Overall, SEMA and Lmer produce very similar estimates of σ^2, although the condition of $n_j = 10, r = 7$, $\text{lvl}_1 = 8$ remains difficult for SEMA even when 50,000 data points have entered.

Figure 5.2 presents the average squared prediction error for each of our two methods. For clarity reasons, only three conditions are presented in Figure 5.2. Note that for both fitting procedures, the error is implemented such that when an individual is entering for the first time, this y_{ij} was predicted using $\bar{y}$. In all conditions (also the three not presented), Lmer had a larger error than SEMA. This is due to two reasons. First, Lmer was often unable to converge in the beginning of the data stream. When Lmer did not converge, the average of the dependent variable ($\bar{y}$) was used to predict the next y_{ij}. Second, while the parameter estimates of SEMA are updated every data point, Lmer was only called once every 1,000 data points. Then, the parameter estimates of Lmer were used for 1,000 data points to predict the next data points. It was computationally infeasible to re-estimate the model using Lmer each time a new data point entered. Likely, in any practical application, this batch type prediction would be chosen if online methods are unavailable.

5.5 SEMA in action: predicting weight fluctuations.

In this section, the SEMA algorithm is applied to an empirical data stream originating from an experiment done by Kooreman and Scherpenzeel (2014). The study

Table 5.1: The variability of the estimates of β, across replications of Lmer and SEMA

					Lmer				SEMA			
n_j	r	lvl_1	β	n	$\hat{\beta}$	SD	2.5%	97.5%	$\hat{\beta}$	SD	2.5%	97.5%
10	2	3	1.5	1,000	–	–	–	–	1.528	0.370	0.827	2.098
				25,000	1.508	0.047	1.415	1.602	1.511	0.066	1.411	1.634
				50,000	1.505	0.037	1.438	1.567	1.507	0.039	1.440	1.591
			-5.5	1,000	–	–	–	–	-5.194	0.321	-5.807	-4.595
				25,000	-5.500	0.034	-5.555	-5.447	-5.499	0.035	-5.559	-5.445
				50,000	-5.499	0.024	-5.542	-5.452	-5.499	0.024	-5.542	-5.452
50	2	3	1.5	1,000	–	–	–	–	1.392	0.279	0.859	1.947
				25,000	1.490	0.069	1.331	1.604	1.490	0.067	1.341	1.601
				50,000	1.491	0.065	1.339	1.591	1.491	0.065	1.343	1.587
			-5.5	1,000	–	–	–	–	-5.292	0.235	-5.783	-4.869
				25,000	-5.497	0.032	-5.554	-5.437	-5.497	0.032	-5.554	-5.437
				50,000	-5.497	0.021	-5.535	-5.461	-5.497	0.021	-5.535	-5.461
10	2	8	1.5	1,000	–	–	–	–	1.495	0.604	0.682	1.877
				25,000	1.500	0.041	1.415	1.570	1.511	0.169	1.405	1.583
				50,000	1.502	0.035	1.432	1.568	1.506	0.061	1.426	1.568
			-5.5	1,000	–	–	–	–	-5.098	0.328	-5.723	-4.394
				25,000	-5.497	0.038	-5.566	-5.424	-5.496	0.038	-5.564	-5.420
				50,000	-5.496	0.026	-5.544	-5.444	-5.496	0.026	-5.544	-5.445
50	2	8	1.5	1,000	–	–	–	–	1.464	0.341	0.839	2.077
				25,000	1.501	0.073	1.361	1.641	1.500	0.073	1.358	1.639
				50,000	1.500	0.071	1.374	1.632	1.501	0.071	1.377	1.637
			-5.5	1,000	–	–	–	–	-5.246	0.280	-5.812	-4.699
				25,000	-5.495	0.035	-5.554	-5.430	-5.495	0.035	-5.554	-5.430
				50,000	-5.497	0.023	-5.542	-5.459	-5.497	0.023	-5.542	-5.459
10	7	8	1.5	1,000	–	–	–	–	1.269	0.848	-0.187	2.772
				25,000	–	–	–	–	1.469	0.238	0.962	1.972
				50,000	1.502	0.039	1.419	1.576	1.494	0.080	1.344	1.657
			-5.5	1,000	–	–	–	–	-3.532	1.025	-5.454	-1.431
				25,000	–	–	–	–	-5.175	0.337	-5.563	-4.305
				50,000	-5.498	0.044	-5.574	-5.411	-5.401	0.127	-5.558	-5.093
50	7	8	1.5	1,000	–	–	–	–	1.355	0.695	-0.244	2.502
				25,000	1.489	0.071	1.371	1.632	1.489	0.070	1.374	1.626
				50,000	1.492	0.073	1.374	1.633	1.491	0.073	1.367	1.626
			-5.5	1,000	–	–	–	–	-3.679	1.077	-5.418	-1.414
				25,000	-5.490	0.083	-5.653	-5.337	-5.475	0.088	-5.630	-5.309
				50,000	-5.487	0.080	-5.644	-5.331	-5.486	0.080	-5.648	-5.325

Note: n_j indicates the condition under which the data streams were generated. This means that the average number of observations per individual is not equal to 10 (50), until $n = 50,000$.

Table 5.2: The variability of the estimates of ϕ, across replications of Lmer and SEMA

					Lmer				Sema			
n_j	r	lvl_1	ϕ	n	$\hat{\phi}^2$	SD	2.5%	97.5%	$\hat{\tau}^2$	SD	2.5%	97.5%
10	2	3	4	1,000	–	–	–	–	15.718	4.337	7.964	22.034
				25,000	4.000	0.211	3.575	4.335	6.468	0.706	4.649	7.379
				50,000	4.001	0.137	3.738	4.258	4.396	0.215	3.989	4.664
			9	1,000	–	–	–	–	9.554	6.051	1.891	23.114
				25,000	9.007	0.374	8.399	9.807	8.057	1.979	3.963	11.041
				50,000	9.018	0.257	8.560	9.580	8.860	0.403	7.875	9.422
50	2	3	4	1,000	–	–	–	–	13.831	2.989	7.856	19.895
				25,000	4.024	0.220	3.636	4.466	4.022	0.219	3.631	4.464
				50,000	4.032	0.205	3.684	4.439	4.032	0.205	3.687	4.441
			9	1,000	–	–	–	–	8.696	5.356	2.111	18.319
				25,000	8.987	0.481	8.220	10.148	8.987	0.480	8.230	10.146
				50,000	8.990	0.444	8.294	9.930	8.990	0.443	8.296	9.930
10	2	8	4	1,000	–	–	–	–	25.898	**23.842**	15.921	48.044
				25,000	4.018	0.201	3.588	4.451	7.196	0.828	6.521	8.209
				50,000	3.999	0.140	3.774	4.283	4.507	0.167	4.276	4.794
			9	1,000	–	–	–	–	14.465	**20.441**	3.124	24.306
				25,000	8.981	0.315	8.391	9.568	9.257	1.775	5.356	11.121
				50,000	8.994	0.224	8.586	9.388	9.013	0.306	8.486	9.528
50	2	8	4	1,000	–	–	–	–	19.502	4.000	13.710	27.796
				25,000	3.982	0.251	3.448	4.454	3.982	0.250	3.441	4.452
				50,000	3.975	0.223	3.562	4.356	3.976	0.223	3.560	4.355
			9	1,000	–	–	–	–	13.512	6.422	4.646	25.029
				25,000	8.921	0.474	7.913	9.762	8.922	0.475	7.917	9.773
				50,000	8.928	0.432	8.101	9.646	8.928	0.432	8.100	9.647
10	7	8	4	1,000	–	–	–	–	17.499	5.998	8.853	29.737
				25,000	–	–	–	–	7.870	1.398	5.723	10.464
				50,000	4.000	0.156	3.714	4.288	4.449	0.451	3.667	5.326
			9	1,000	–	–	–	–	9.320	6.968	2.201	24.281
				25,000	–	–	–	–	6.358	2.529	2.434	12.214
				50,000	8.971	0.279	8.465	9.453	6.855	1.205	4.656	9.175
50	7	8	4	1,000	–	–	–	–	15.896	5.963	7.317	31.372
				25,000	3.971	0.236	3.546	4.436	3.965	0.238	3.536	4.439
				50,000	3.959	0.197	3.566	4.325	3.960	0.197	3.567	4.323
			9	1,000	–	–	–	–	11.417	12.801	2.597	35.083
				25,000	8.999	0.441	8.165	9.857	8.996	0.442	8.164	9.873
				50,000	8.999	0.417	8.310	9.745	8.999	0.417	8.309	9.746

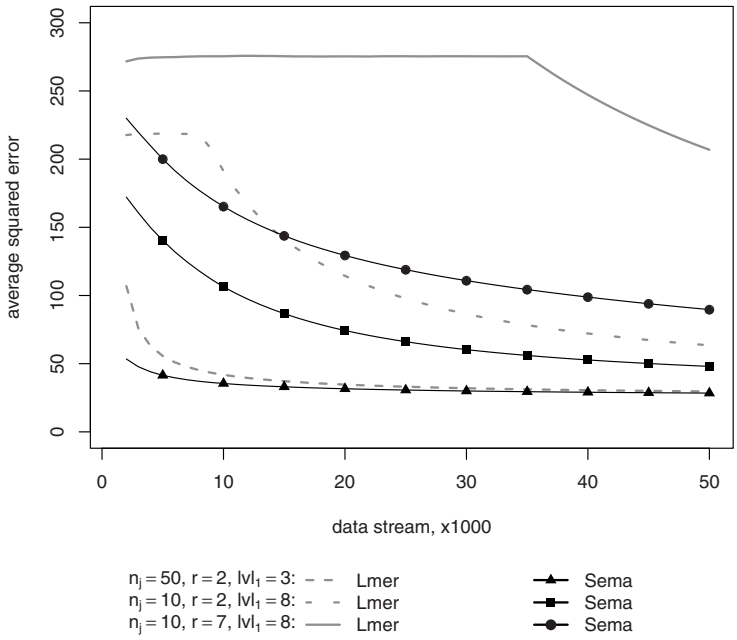


Figure 5.2: Average squared prediction error of three selected conditions. When Lmer failed to fit the model, the $\bar{y}$ was used to predict y_{ij}. When Lmer returned model parameters, these model parameters were used to predict the next 1,000 data points, after which the model was fitted again to update the model parameters.

Table 5.3: The variability of the estimates of σ^2 across replications of Lmer and SEMA

				Lmer				SEMA			
n_j	r	lvl_1	n	$\hat{\sigma}^2$	SD	2.5%	97.5%	$\hat{\sigma}^2$	SD	2.5%	97.5%
			1,000	–	–	–	–	21.632	3.318	16.553	28.633
10	2	3	25,000	24.936	0.254	24.548	25.551	24.659	0.911	23.657	26.977
			50,000	24.985	0.178	24.627	25.308	24.921	0.195	24.568	25.314
			1,000	–	–	–	–	22.856	2.624	18.312	28.203
50	2	3	25,000	25.068	0.206	24.727	25.534	25.069	0.206	24.726	25.528
			50,000	25.012	0.167	24.679	25.348	25.012	0.167	24.679	25.349
			1,000	–	–	–	–	28.568	**21.752**	20.013	47.730
10	2	8	25,000	25.019	0.288	24.470	25.531	24.202	0.524	23.429	25.512
			50,000	24.985	0.179	24.619	25.350	24.865	0.175	24.494	25.269
			1,000	–	–	–	–	25.095	3.211	20.387	33.003
50	2	8	25,000	25.020	0.239	24.566	25.449	25.021	0.240	24.565	25.452
			50,000	25.006	0.155	24.726	25.293	25.006	0.155	24.725	25.293
			1,000	–	–	–	–	33.778	10.041	19.710	56.259
10	7	8	25,000	–	–	–	–	39.729	5.599	30.297	50.673
			50,000	25.001	0.217	24.635	25.397	29.656	2.173	26.032	34.261
			1,000	–	–	–	–	38.022	10.165	20.874	57.999
50	7	8	25,000	25.027	0.238	24.622	25.498	25.030	0.236	24.630	25.496
			50,000	25.016	0.159	24.737	25.345	25.016	0.159	24.739	25.345

concerned the fluctuations in individuals' weight—over repeated measurements—in a longitudinal study using respondents from the Longitudinal Internet Studies for Social Sciences (LISS) panel. Among the respondents of the LISS panel, about 1,000 *smart scales* were handed out. These smart weighting scales were equipped with an Internet connection. Respondents were instructed to use the scale barefoot, such that it could measure, among other variables, weight, percentage of muscle tissue, and percentage of fat tissue. The smart scale sent the data to a central LISS server, where the data were combined with respondents' survey data. The smart scales were handed out in the beginning of 2011 and the data collection continued until February 2014. While the data set contains the data from roughly 3 years, the authors used the data of 2011 only. Because the data of the smart scales includes time stamps, we were able to replay the data stream from 2011 till February 2014. Thus, in this evaluation of SEMA, the data of Kooreman and Scherpenzeel ($n = 78{,}021$, $N = 883$) were combined with the data of the remaining years. The first experimental factor was the (instructed) frequency of the scale usage: every day, every week, or not specified. The second factor was the feedback respondents received: their weight and the norm what they should weigh, their weight and their goal weight, or only their weight. Both experimental factors were crossed, resulting in nine conditions. Finally, we removed a number of outliers (0.1% of the data), for which weight fluctuated with more than 5 kg within a day for a single respondent. The remaining data set consisted of $n = 288,521$ observations from a total of $J = 1,269$ respondents. Table 5.4 presents an overview of the model fitted to the data stream by indicating the variables included as fixed or random, as well as the number of levels (or categories) of each of the variables.

We analyzed the data stream again offline using the Lmer function and online

Table 5.4: Fitted model to the smart-scale data stream

Variables	Fixed	Random	number of categories	Reference
Day of the week	✓	✓	7	Friday
Gender	✓		2	male
Year of birth	✓		–	1970 (centered)
Length	✓		–	174cm (centered)
Feedback	✓		3	only weight
Frequency	✓		3	not specified
Time of Measurement	✓		4	morning

The dependent variable is *weight*

Starting values:
fixed effect intercept $= 84$,
fixed effect Gender $= 14$,
variance random intercept $= 100$,
remaining fixed-effects parameters started at 0, the variance of the random effects at 1

using the SEMA algorithm. Lmer was again called every $n = 1,000$ to update the parameter estimates. In addition to the implementation of SEMA as introduced in this chapter, we included three extra implementations of SEMA which have been evaluated by Ippel et al. (2016b) for the random-intercept model:

- SEMA Training: this implementation includes a training-data set to obtain good starting values,
- SEMA Update: this implementation includes extra full E-steps to recompute all individuals' contributions to the CDSS at given intervals,
- SEMA Training and Update: this implementation is a combination of the two above.

The SEMA Training implementation used the first $n = 5,000$ data points as a training set to obtain good guesses for the parameter starting values (using the traditional offline EM algorithm). The SEMA algorithm as presented in this chapter then used these starting values to continue the analysis of the data stream. The second implementation, SEMA Update, is similar to the SEMA algorithm as presented in this chapter, though in addition to the E-steps per individual, SEMA Update recomputed the CDSS at given intervals by performing a single full E-step for all individuals followed by an M-step, as opposed to computing the E-step only for the newly arriving individual. The last implementation, SEMA Training and Update, is a combination of the two previous implementations by using both a training set as well as additional full E-steps. In a practical setting, when starting values are difficult to choose, using the beginning of the data stream as a training set limits number of steps SEMA has to take to obtain good estimates of the parameters. In addition, the SEMA algorithm corrects the previous contributions of an individual to the CDSS. However, in this data stream we do not know if the individual steps on the smart scale again. By evaluating all individuals at given intervals ($n = 1,000$), previous contributions to the CDSS can be updated, even though an individual has not returned (yet). So,

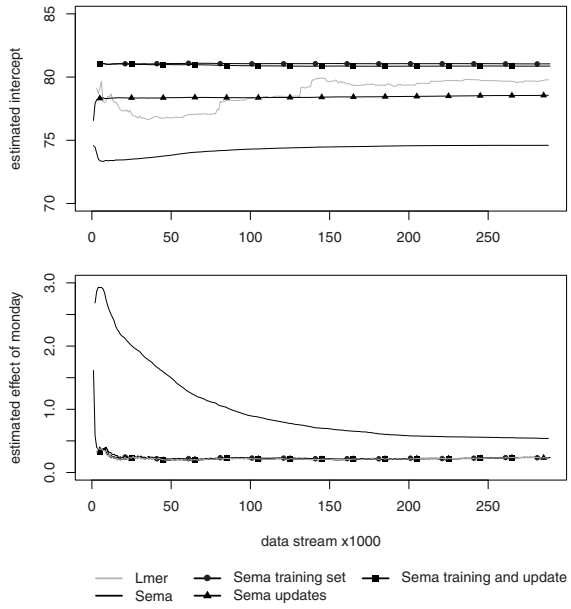


Figure 5.3: Estimated intercept and the effect of Monday, fixed effects

using the additional updates, the SEMA algorithm can correct the contributions to the CDSS in a computationally efficient manner. Below the results of all five fitting procedures are presented.

Since the authors of the original analysis focused on the "effect of Monday", which implies that on average individuals where 0.2 kg heavier on Mondays than on Fridays, we also focused on the estimation of this "Monday" effect. Figure 5.3 presents the estimated fixed intercept of the model in the top figure, and the average effect of Monday in the bottom figure. The x-axes present the length of the data stream and the y-axes the estimated parameter values. In gray are the parameter estimates obtained using the offline procedure. These are clearly more fluctuating than the SEMA procedure(s), which is what can be expected given that Lmer is run till convergence. SEMA, without extra EM runs or using a training set, underestimated the intercept and overestimates the effect of Monday due to the selected starting values. When a training set is used – or when additional E-steps are included – Figure 5.3 shows that the estimates resulting from SEMA are accurate while being magnitudes faster than the traditional procedure.

Figure 5.4 illustrates the estimated variances of the random intercept and the variance of the effect of Monday. In both graphs, SEMA again overestimated the variance due to the selected starting values, while the other fitting procedures were very similar: $\hat{\phi} = 0.095$, (Lmer), 1.269 (SEMA), 0.164 (SEMA Training), 0.090 (SEMA Update), and 0.071 (SEMA Training and Update). Interestingly, all methods, during pretty much all of the data stream, estimate the variance of the effect of Monday

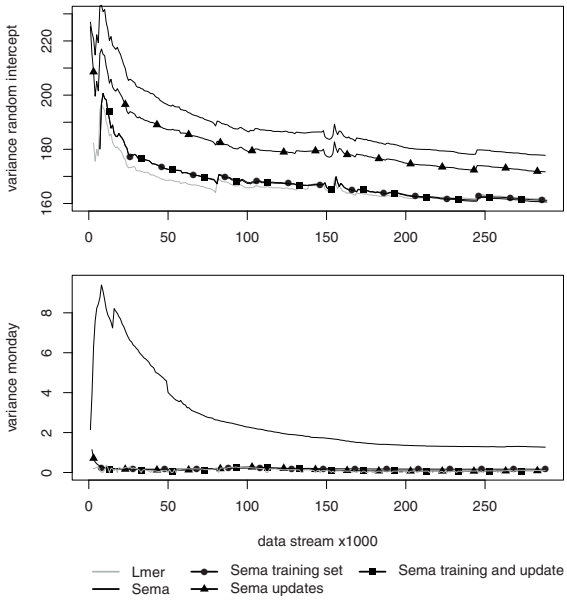


Figure 5.4: Estimated intercept and the effect of Monday, random effects

to be high compared to its average effect. To us, this indicates that the "Monday effect" might be less systematic than emphasized in the original study: apparently, for quite a large number of individuals, the effect of Monday is negative as opposed to positive.

Finally, Figure 5.5 presents the average squared prediction error of all five fitting procedures. The average squared prediction error was implemented similar to the simulation study, where the average weight was used to predict the weight of an individual which entered for the first time. The average of the squared prediction error is computed from $n = 6,000$ onwards, because the offline procedure could estimate the model from $n \approx 5,000$. We disregarded the beginning of the stream for all fitting procedures and compared the methods only from a point at which they produce parameter estimates. While computing the error only after Lmer has converged favors the offline procedure, still Lmer produces on average more prediction error than SEMA. Hence, it seems that for the purpose of predicting individual-level effects in data streams, SEMA is very well suited.

5.6 Discussion

In this chapter, we developed an extension of the Streaming Expectation Maximization Approximation (SEMA) algorithm (Ippel et al., 2016b). This extension enables researchers to fit multilevel models that include fixed effects at level 1 (i.e. repeated

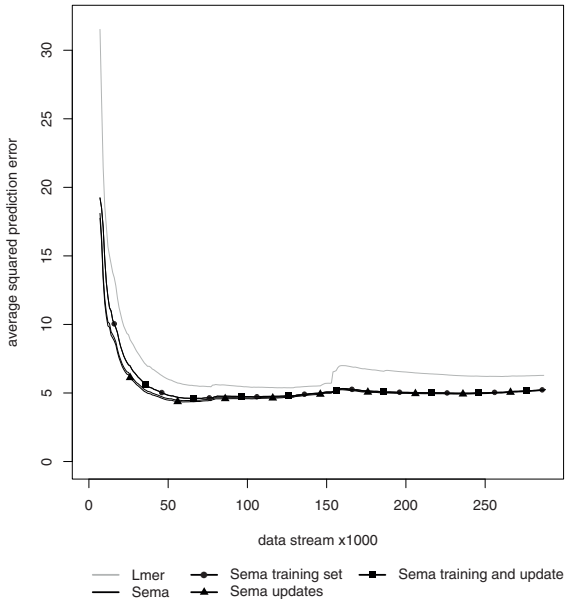


Figure 5.5: Average squared prediction error of weight, starting from $n = 6,000$

measurements), level 2 (i.e., individuals), and random intercepts and random slopes when data sets are large and/or continuously augmented. SEMA is computationally more efficient because this algorithm never revisits older data points.

Common procedures to fit multilevel models (i.e., EM algorithm or Newton Raphson) repeatedly pass over the data to estimate the model parameters. When new data enter, these procedures revisit all data points to update the model parameters. Especially when the number of random effects increases and many passes over the data are required to obtain stable estimates of the model parameters, these traditional fitting procedures quickly become infeasible for large data sets or continuous data streams. SEMA, on the other hand, only uses a data point once, after which it is discarded. SEMA thus learns the maximum likelihood values of the model parameters more efficiently in a data stream than the common procedures, since the model parameter estimates are updated with each newly entered data point. Therefore, SEMA facilitates the analysis of data streams while accounting for the nested structure that is often observed in data streams. SEMA effectively deals with the problems of storing extremely large data sets and very lengthy fitting procedures. Our algorithm enables researchers to use multilevel models for prediction in real time. In a simulation study, we showed that even when the number of observations per individual is small and the number of parameters is large, parameter estimates were estimated quite accurately. Furthermore, we showed that the predictive performance of SEMA was competitive, if not superior, to current state-of-the-art methods.

While the current extension of SEMA algorithm allows for fitting multilevel models with fixed and random effects in data streams, extensions are possible and need further development. First, in Kooreman and Scherpenzeel (2014)—our empirical example—the authors actually used a multilevel model with more predictors than the model we used in this chapter. The original model also contained fixed effects for the calendar months. Fitting this model requires observations in (almost) each month, such that the $\boldsymbol{X}'\boldsymbol{X}$ matrix becomes invertible (i.e., at least semi positive definite). Consequently, fitting a model including the effects of months cannot be fitted to the data *before* the data stream has run for almost a year. Further research should focus on extending the model *during* the data stream, such that these effects can be included dynamically once enough data has been collected.

Second, the current version of SEMA basically assumes that the true data generating process is stationary and that, over the course of the data stream, we converge to the "correct" parameter estimates. However, when monitoring individuals over time, it is likely that the data-generating process itself changes over time. Moving window approaches, in which only the latest data points are included in the analysis, are often used in such cases. However, when using a moving window approach one would still refit the model to all the data points in the window every time the window changes. Alternatively, however, we could introduce a fixed learn rate when dealing with data streams. In Eq. 5.1 it is easily seen that the "learn-rate" for computing an online sample mean is $\frac{1}{n}$. Thus, as the stream becomes longer (and n grows larger) the learn rate decreases and the computed mean stabilizes. If instead we would alter the update rule of $\bar{x}$ to read $\bar{x} := \bar{x} + \frac{x_t - \bar{x}}{\texttt{min}(n,\alpha)}$ for some fixed value of α of say $10,000$, we effectively create a moving window in the sense that older data points are *smoothly* discarded—though without revisiting older data points. This can, with some effort, similarly be implemented in SEMA.

To conclude, our extended SEMA algorithm is a computationally-efficient algorithm to analyze data that contain a nested structure and arrive in a continuous fashion. Hence, multilevel models with numerous fixed and random effects can now be fit to continuous data streams (or extremely large static data sets), in a computationally efficient fashion.

Acknowledgements

We would like to express our thanks to prof. Peter Kooreman for sharing their data. Furthermore, we thank Daniel Ivan Pineda for his feedback on the writing process.

Chapter 6

Discussion

6.1 Overview

In this thesis, computationally-efficient procedures for analyzing data streams are developed and illustrated in order to make data streams more accessible to applied researchers. In the case of independent observations, Chapter 2 gives a short overview of approaches to analyze streaming data. A large part of this chapter is dedicated to the introduction of online learning methods. These methods are illustrated with examples written in R code to promote the use of these online learning methods.

Data streams, however, often consist of repeated measurements of the same individuals, which creates a nested structure in the data. Chapter 3 presents four online learning methods to deal with dependent observations in data streams where the outcome variable is binary. These online learning methods are based on shrinkage factors which are well-known in the literature. Additionally, this chapter introduces an online learning method to deal with the normality assumption based on Morris and Lysy's (2012) data transformation for binary outcomes.

In Chapter 4, a novel online algorithm is developed, called the Streaming Expectation Maximization Approximation algorithm (SEMA). This algorithm fits random intercepts models on streaming data. Based on the EM algorithm (Dempster et al., 1977), SEMA fits the simplest multilevel model by updating the parameter estimates online. This online learning approach to multilevel modeling facilitates social scientists to study social phenomena in a longitudinal design using data streams.

Lastly, in Chapter 5 an extension of SEMA is provided. This extension can be used to estimate more complex multilevel models, which include fixed effects and random intercepts and slopes. The development of these methods allows researchers to broaden the scope of their research using data streams.

6.2 Related approaches to analyze data streams

In this thesis, I have predominantly explored the online learning approach as a method to efficiently analyze data streams. In addition to online learning, two related approaches have been discussed in Chapter 2: a sliding window approach and

a parallelization approach. The sliding window approach reduces memory burden because it only stores the most recent set of data points. Using this technique, previous data points are discarded from memory when new data points enter (Aggarwal, 2007; Gaber et al., 2005). Parallelization, on the other hand, decreases the computational burden by distributing the analyses over multiple machines. This technique is commonly used for large static data sets (Chu et al., 2006). Some strong aspects of both techniques could also be implemented in the online learning methods developed in this thesis.

6.2.1 Sliding window approach

As mentioned above, the sliding window approach only uses the most recent set of observations. An often unintended practical advantage of a sliding window is that when changes occur over time, this approach can easily follow these changes, because the older data points do not influence the analyses anymore. Let us use the sample mean as an example. When a new data point enters the oldest data point is deleted from memory to include the new data point and the sample mean is recomputed over this 'new' set of observations. This set always consists of the same number of observations, m, independent of the length of the data stream.

On the other hand, computing the sample mean online as more data enter, the weight of the new data points decreases gradually. The weight of a new observation equals $\frac{1}{n}$, where n is the total number of observations and when the data grows larger, this weight becomes smaller. In this case, the learn rate of the online mean ($\frac{1}{n}$) becomes smaller when more data enter. As a result, the value of the mean hardly changes anymore when n becomes large. If, however, the learn rate is fixed to a value larger than $\frac{1}{n}$, a smooth 'sliding window' is created. In this smooth sliding window, the older data points are not ignored, though the weight of these observations becomes smaller compared to the new observations.

Fixing learn rates is also an interesting future direction for the online learning methods discussed in this thesis. For instance, in Chapter 5, when many data points stream in, the SEMA algorithm did not learn much anymore from the new observations. Including a fixed learn rate would make SEMA more sensitive to new observations. As a result, the parameter estimates become more variable.

Usually, it is undesirable to have highly fluctuating parameter estimates because these are difficult to interpret. However, when the parameter estimates are perfectly stable, adding new data becomes rather useless. This balance is related to the bias-variance trade off (see for an introduction G. James, Witten, Hastie, & Tibshirani, 2013). In this trade off, bias is related the robustness of the parameter estimate against random fluctuations in the data. The variance is related to the sensitivity to pick up real changes in the data. In practice, some balance between bias and variance should be found to obtain accurate predictions.

6.2.2 Parallelization

Parallel computing is useful in cases where the computations of an analysis can be split up in separate blocks. To come back to the example of the sample mean: each machine could compute the sample mean of the data stored on that machine, next, all these (sub)means are combined to compute the mean over all data. The combination of parallelization and online learning is known in the literature (Böse, Andrzejak, & Högqvist, 2010; Hsu, Karampatziakis, Langford, & Smola, 2011).

The computations of the online shrinkage factors and the SEMA algorithm can easily be distributed over multiple machines by making use of the nesting in the data. For instance, let us assume there are five machines and 100 individuals are repeatedly observed over time. Four machines could be used to store the summary statistics of all the individuals. The fifth machine stores the global parameters, such as the mean over all data and a list of which individual is stored on which machine. Now, when new data enter, the fifth machine retrieves the record of the individual that just entered from one of the four machines, updates the global parameters and the individual summary statistics, which can be returned to the right machine. Distributing these records reduce the memory burden of the introduced online methods.

6.2.3 Bayesian framework

In this thesis, I have focused primarily on the Maximum Likelihood approach to estimate parameters. An alternative, Bayesian, approach to parameter estimation is obtained by quantifying one's prior beliefs regarding the values of the parameters using a probability distribution (or density in the continuous case), called the prior distribution ($p(\theta)$). Next, using the likelihood of the data ($p(D|\theta)$) and Bayes Theorem, one's prior beliefs can be updated, resulting in the so-called posterior: $p(\theta|D) \propto p(D|\theta)p(\theta)$ (see, e.g., Gelman et al., 2004; Lynch, 2007; van de Schoot et al., 2014). Theoretically, this scheme of updating one's belief lends itself very well to analyzing data streams because the posterior after observing a new data point y_{t+1} is proportionally equal to the likelihood of the new observation multiplied with the posterior based on the previous data points (Opper, 1998): $p(\theta|D_{t+1}) \propto p(y_{t+1}|\theta)p(\theta|D_t)$.

While the Bayesian approach fits conceptually well with data streams, the reality is, however, less straightforward. Even when the data do not enter in a stream, many Bayesian models are computationally complex because the posterior does not always have a known distribution. In practice, researchers often rely on techniques such as Markov Chain Monte Carlo sampling (MCMC) to approximate the posterior. In such cases, updating the posterior in the context of data streams could be computationally challenging.

Interesting developments in Bayesian updating are variational methods and sequential MCMC (sMCMC) sampling. Variational methods speed up the updating process by replacing the posterior which has an unknown distributional form by a distribution with a known distributional form (Broderick, Boyd, Wibisono, Wilson,

& Jordan, 2013; Kabisa (Tchumtchoua), Dunson, & Morris, 2016). The other development, sMCMC, provides an appealing extension to popular MCMC methods which is computationally attractive since the generated MCMC draws are updated as opposed to sampled anew when additional data enters (Yang & Dunson, 2013). The SEMA approach presented in this thesis, which involves updating the likelihood during a data stream, might be relevant to this field of research as well since evaluation of the likelihood is necessary if one wishes to update the posterior.

6.3 Data stream challenges

6.3.1 Convergence

First, the exact meaning of convergence is less clear in the context of data streams compared to static data. In practice, when analyzing static data sets, an algorithm might repeatedly go over the data to find the optimum of a function. The parameter values are compared over consecutive iterations and if these values change less than a given threshold, the algorithm converged. When the function is not convex, an algorithm might converge to a local optimum instead. A multi-start procedure might help to find the global maximum.

A multi-start procedure is also possible in a data stream, by running the same analysis started at different points in parallel. However, in the context of data streams, it is no longer feasible to repeatedly go over the data to find an (global or local) optimum. Additionally, due to the influx of new data, both the likelihood of the data given the current parameter estimates and the parameter estimates themselves change. These two reasons of why the likelihood changes, complicate studying convergence in a data stream.

In the context of static data, one hopes to find stable parameter estimates, as an indication that the algorithm converged. However, when analyzing data streams, stable parameter estimates are not necessarily desirable; if due to a shift of the data generating model, data enter that do not fit well with the current parameter estimates, parameter estimates should change accordingly. The developed online methods do not 'forget' older data and it is assumed that the data generating model is stable. As a result, when the data generating model changes over time, i.e., the strength or direction of an effect changes, the parameter estimates will be influenced by the previous data generating model. Introducing a forgetting factor, such as described in Section 6.2.1, could decrease this influence in the parameter estimates.

Even though parameter values are likely to change in a data stream, it is of interest to see whether the current parameter values are close to the Maximum Likelihood values of the data seen so far. Theoretical convergence of online versions of EM algorithm have been studied (Cappé & Moulines, 2009; Neal & Hinton, 1998), though currently, I have not extensively explored a formal procedure to check whether SEMA has converged. In Chapter 4 a heuristic procedure is used to monitor

whether the parameter estimates become more stable over time. Studying convergence is an interesting direction for future research.

6.3.2 Models used for analyses

Choosing a model for the analysis is also more complex when analyzing data streams instead of static data sets. For instance, one might discover during the data stream that the fitted model has to be altered; e.g., excluding current covariates and/or including other variables. Firstly, adapting models during the stream is only possible if the required information is stored: information which is not stored, might not be retrievable. Secondly, even if the information is available, including more variables during the stream might complicated. The application study in Chapter 5 provides a good example of such a situation. In this application, individuals' weight was monitored over three years. The original authors of this study controlled for the effect of which month the data were collected. Fitting that same model in a data stream, would require observations in each month. However, waiting for all months to be observed is usually not an option. For such cases, an approach to include covariates while the data are entering should be developed.

In the presented simulation studies, the models which generated the data were also fitted to the data stream. In this thesis, I did not explicitly explore the robustness of methods, when a different model was fit to the data or how the fitted model could be adapted during the data stream. When analyzing static data, different models can be fitted to the data and compared which model fits best using goodness-of-fit measures, e.g., AIC or BIC. An option for data streams could be to fit multiple models in parallel and decide later which model is preferred. However, evaluating which model is best, is less straightforward in a data stream than in static data sets. The AIC and BIC, commonly used to compare different models, are based on the log likelihood, which is computed using the current parameter estimates. In a data stream, the parameter estimates are continuously updated and especially in the beginning of the data stream probably far from the Maximum Likelihood solution. As a result, the differences in AIC or BIC values between the competing models could be an artifact of poorly-chosen starting values, which makes comparing models in a data stream using AIC and BIC complex.

An interesting direction for future research to compare competing models in data streams could be inspired by the SEMA algorithm: when a new data point enters, SEMA updates the contribution of the individual belonging to this data point to the complete data sufficient statistics. These sufficient statistics are in fact summations over individuals. Since the log likelihood is a sum of individual log-likelihood contributions, these contributions could also be updated when the individuals return in the data stream. Such an updating scheme could make it possible to employ an online approximation of common goodness-of-fit measures in data streams.

6.3.3 Missingness

Especially when a model contains many covariates, it is likely that not all covariates are observed for each data point. Let us assume $i = 1 \dots n$ data points and each data point consists of k covariates. In the case of data streams, in addition to not observing all k covariates for each data point, some covariates might not enter all at the same time. Some information of the covariates could be potentially be retrieved from memory, for instance a level 2 covariate like gender. However, other information might be missing or only drop in later, e.g., learning the gender of the respondent later in the data stream. While missingness is a research area on its own (Donders, van der Heijden, Stijnen, & Moons, 2006; van der Palm, van der Ark, & Vermunt, 2016), dealing with missingness in a data stream complicates the issue of having incomplete observations even further. In this thesis, it is assumed that for observation i, the data of all k covariates are observed at once. Dealing with covariates that do not enter at the same time or do not enter at all, is an interesting research area, though not studied in this thesis.

6.3.4 Attrition

Lastly, a challenge common for longitudinal research in general is attrition, i.e., respondents quit participating in the study. Attrition is a threat because it can affect the generalizability of the sample. If a subgroup of respondents, e.g., the less affluent respondents, drop out of the study, the parameter estimates of the model could become biased. The methods developed in this thesis are even more affected by attrition due to the update scheme of the parameter estimates. The estimates of shrinkage factors (Chapter 3) and the parameters of the multilevel models in Chapter 4 and 5 are updated in two steps: first the previous contributions of this individual is subtracted of the summary statistics of the parameters, second the new contribution is added to the summary statistics. The parameter estimates are updated with each new data point. However, these individual contributions to these summary statistics are only updated when an individual returns. So, except for the individual who just entered, all the remaining contributions are still based on the 'outdated' parameter estimates. As a result, when an individual does not return in the data stream, its outdated contributions are not updated with the new parameter estimates.

While solutions to attrition in data streams have not been extensively studied in this thesis, two approaches have been implemented. First, in Chapter 3, the contributions of those respondents which had indicated to stop participating in the panel study, were removed from the summary statistics. However, this information of when a respondent stops participating is not always available. The second approach is applied in Chapter 4 and 5. In these chapters, the SEMA algorithm did additional runs over all the individuals at given intervals. During these additional runs, the contributions of all individuals were updated, including those who had not returned in the data stream. Alternatively, one could also update the contributions of those

who dropped out or do not return that often, when individuals similar to them do return in the data stream (which is related to partial EM algorithm, Neal & Hinton, 1998; Thiesson et al., 2001).

6.4 Null Hypothesis Significance Testing

Whereas the methods discussed in this thesis can be used to make decisions sequentially while the data are entering, these techniques are *not* intended to sequentially *test* statistical hypotheses. This issue has also been touched upon in Chapter 2. It is considered a Questionable Research Practice when the data collection is stopped based on whether the p-value is small enough (Simmons et al., 2011). When the data collection is stopped based on the outcome of the null hypothesis test, invalid conclusions could be drawn based on the test results: the probability of obtaining a p-value smaller than the chosen Type I error rate without the effect being present in the population could be severely inflated.

Besides the inflated Type I error issue, given the research context is one of either extremely large data sets or data streams which often result in large data sets, the usefulness of the p-value is questionable. The size of the p-value is related to the effect size (e.g., the observed difference between two groups) and the sample size. When the effect size increases, the p-value becomes smaller, given that the sample size remains the same. On the other hand, when the sample size increases and the effect size remains the same, the p-value also becomes small, even though the effect size is practically not meaningful. It is, therefore, preferable to focus on effect sizes instead of p-values (Wilkinson & Task Force on Statistical Inference, 1999).

Even when the focus is on effect sizes, it is of interest to get some insight in the variance of the estimate of the effect size. A commonly-used approach to estimate the uncertainty of an estimate is a bootstrapping procedure. Using bootstrapped samples, standard errors can be estimated (McLachlan & Krishnan, 2007). To estimate standard errors in data streams, computationally-efficient bootstrap procedures are available (e.g., Owen & Eckles, 2012). An online bootstrap procedure supplementing the developed methods provides more insight of the certainty of the obtained effect sizes.

6.5 Future research directions for SEMA

In this chapter, various directions for future research in data streams were presented. In Section 6.2, it was illustrated how other commonly-used techniques for the analysis of data streams could be implemented and improve the online methods developed in this thesis. In Section 6.3 challenges of analyzing data streams and potential directions for solutions were discussed. In this last section, several future directions specifically for SEMA are presented.

While SEMA is extended in Chapter 5 to fit linear multilevel models with fixed and random effects, other model extensions yet have to be developed. For example, SEMA as described in this thesis, cannot fit a model with more than two levels. As an illustration let us return to the baseball example from Chapter 1: a two-level model could be the batting observations nested within the baseball players; the third level in this example are the teams in which the baseball players are nested. Another extension of the SEMA algorithm could be a crossed random effects model. This model assumes that the observations are nested within more than one grouping, whereby the groupings themselves are not nested: observations nested within customers and the same observations are also nested within different webpages.

Finally, the SEMA algorithm currently fits *linear* multilevel models. The linear multilevel models are part of a larger framework of multilevel models, namely the Generalized Linear Mixed Models (GLMM, Skrondal & Rabe-Hesketh, 2004). The GLMM framework also contains multilevel models for variables which are not continuous. However, if the outcome variable is categorical, model fitting could be severely complicated because the likelihood function can often not be straightforwardly maximized (Bock & Aitkin, 1981; Breslow & Clayton, 1993), which is true even in static data sets. Chapter 3 presents some heuristic online methods to deal with binary outcomes in the context of data streams with nested observations and it might be interesting to explore whether these can be converted into full multilevel models.

References

Aggarwal, C. C. (Ed.). (2007). *Data streams: Models and algorithms*. Springer US. doi: 10.1007/978-0-387-47534-9

Agresti, A. (2002). *Catagorical Data Analysis* (2nd ed.). Wiley series in probability and statistics. doi: 10.1002/0471249688

Agresti, A., Booth, J. G., Hobert, J. P., & Caffo, B. (2000). Random-effects modeling of categorical response data. *Sociological Methodology*, *30*(1), 27–80. doi: 10.1111/0081-1750.t01-1-00075

Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., & Taha, K. (2015). Efficient machine learning for big data: A review. *Big Data Research*, *2*(3), 87 – 93. (Big Data, Analytics, and High-Performance Computing) doi: 10.1016/j.bdr.2015.04.001

Anderson, C. J. (2000). Economic voting and political context: a comparative perspective. *Electoral Studies*, *19*(2-3), 151–170. doi: 10.1016/s0261-3794(99)00045-1

Armitage, P., McPherson, C., & Rowe, B. (1969). Repeated Significance Tests on Accumulating Data. *Journal of the Royal Statistical Society. Series A (General)*, *132*(2), 235–244. doi: 10.2307/2343787

Armstrong, R. A. (2014). When to use the Bonferroni correction. *Ophthalmic & physiological optics*, *34*(5), 502–508. doi: 10.1111/opo.12131

Arvas, E., & Sevgi, L. (2012). A Tutorial on the Method of Moments. *Antennas and Propagation Magazine, IEEE*, *54*(3), 260–275. doi: 10.1109/MAP.2012.6294003

Atallah, M. J., Cole, R., & Goodrich, M. T. (1989). Cascading Divide-and-Conquer: A Technique for Designing Parallel Algorithms. *SIAM Journal on Computing*, *18*(3), 499–532. doi: 10.1137/0218035

Barrett, L. F., & Barrett, D. J. (2001). An Introduction to Computerized Experience Sampling in Psychology. *Social Science Computer Review*, *19*(2), 175–185. doi: 10.1177/089443930101900204

Barrett, P., Zhang, Y., Moffat, J., & Kobbacy, K. (2013). A holistic, multi-level analysis identifying the impact of classroom design on pupils' learning. *Building and Environment*, *59*, 678–689. doi: 10.1016/j.buildenv.2012.09.016

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, *67*(1), 1–48. doi: 10.18637/jss.v067.i01

Beck, E. N. (2015). The Invisible Digital Identity: Assemblages in Digital Networks. *Computers and Composition*, *35*, 125–140. doi: 10.1016/j.compcom.2015.01.005

Berlinet, A. F., & Roland, C. (2012). Acceleration of the EM algorithm: P-EM versus epsilon algorithm. *Computational Statistics & Data Analysis*, *56*(12), 4122–4137. doi: 10.1016/j.csda.2012.03.005

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., … Wiswedel, B. (2009). KNIME – The Konstanz Information Miner. *ACM SIGKDD Explorations Newsletter*, *11*(1), 26–31. doi: 10.1145/1656274.1656280

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive Online Analysis. *The Journal of Machine Learning Research*, *11*, 1601–1604. doi: 10.1.1.180.8004

Bock, R. D., & Aitkin, M. (1981). EM Solution of the Marginal Likelihood Equations. *Psychometrika*, *46*(4), 443–459. doi: 10.1007/BF02293801

Böse, J.-H., Andrzejak, A., & Högqvist, M. (2010). Beyond online aggregation: Parallel and incremental data mining with online map-reduce. In *Proceedings of the 2010 workshop on massive data analytics on the cloud – mdac '10* (pp. 3:1–3:6). New York, USA: ACM. doi: 10.1145/1779599.1779602

Bottou, L. (1999). On-line learning and stochastic approximations. In D. Saad (Ed.), *On-line learning in neural networks* (pp. 9–42). Cambridge University Press. doi: 10.1017/cbo9780511569920.003

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of the 19th international conference on computational statistics (compstat'2010)* (pp. 177–187). doi: 10.1007/978-3-7908-2604-3_16

Breslow, N. E., & Clayton, D. G. (1993). Approximate Inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association*, *88*(421), 9–25. doi: 10.2307/2290687

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., & Jordan, M. I. (2013). Streaming variational bayes. In *Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013. proceedings of a meeting held december 5-8, 2013, lake tahoe, nevada, united states.* (pp. 1727–1735).

Browne, W., & Goldstein, H. (2010). MCMC sampling for a multilevel model with nonindependent residuals within and between cluster units. *Journal of Educational and Behavioral Statistics*, *35*(4), 453–473. doi: 10.3102/1076998609359788

Buskirk, T. D., & Andrus, C. (2012). Smart Surveys for Smart Phones: Exploring Various Approaches for Conducting Online Mobile Surveys via Smartphones. *Survey Practice*, *5*(1), 1–11.

Cappé, O. (2011a). Online EM Algorithm for Hidden Markov Models. *Journal of Computational and Graphical Statistics*, *20*(3), 1–20. doi: 10.1198/jcgs.2011.09109

Cappé, O. (2011b). Online Expectation-Maximisation. In K. Mengersen, M. Titterington, C. Robert, & P. Robert (Eds.), *Mixtures: Estimation and applications* (pp. 31–53). John Wiley & Sons, Ltd. doi: 10.1002/9781119995678.ch2

Cappé, O., & Moulines, E. (2009). On-line expectation-maximization algorithm for latent data models. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *71*(3), 593–613. doi: 10.1111/j.1467-9868.2009.00698.x

Carmona, C. J., Ramírez-Gallego, S., Torres, F., Bernal, E., Del Jesus, M. J., & García, S. (2012). Web usage mining to improve the design of an e-commerce website: OrOliveSur.com. *Expert Systems with Applications*, *39*(12), 11243–11249. doi: 10.1016/j.eswa.2012.03.046

Cheng, H., & Cantú-Paz, E. (2010). Personalized click prediction in sponsored search. In *Proceedings of the third acm international conference on web search and data mining - wsdm '10* (pp. 351–360). New York, USA: ACM. doi: 10.1145/1718487.1718531

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2006). Map-reduce for machine learning on multicore. In *Proceedings of the 19th international conference on neural information processing systems* (pp. 281–288). Cambridge, MA, USA: MIT Press.

Cortes, C., Fisher, K., Pregibon, D., Rogers, A., & Smith, F. (2000). Hancock: A Language for Extracting Signatures from Data Streams. In *Proc. of the sixth acm international conference on knowledge discovery and data mining (sigkdd)* (pp. 9–17). Boston.

Curtin, R., Singer, E., & Presser, S. (2007). Incentives in Random Digit Dial Telephone Surveys: A Replication and Extension. *Journal of Official Statistics*, *23*(1), 91–105.

Datar, M., Gionis, A., Indyk, P., & Motwani, R. (2002). Maintaining Stream Statistics over Sliding Windows. *SIAM Journal on Computing*, *31*(6), 1794–1813. doi: 10.1137/S0097539701398363

Demchenko, Y., Grosso, P., De Laat, C., & Membrey, P. (2013). Addressing big data issues in Scientific Data Infrastructure. *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, 48–55. doi: 10.1109/CTS.2013.6567203

Demidenko, E. (2004). *Mixed models: Theory and applications*. Wiley Series in Probability and Statistics. doi: 10.1002/0471728438

Dempster, A. P., Laird, N. M., & Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.

Donders, A. R. T., van der Heijden, G. J., Stijnen, T., & Moons, K. G. (2006). Review: A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, *59*(10), 1087–1091. doi: 10.1016/j.jclinepi.2006.01.014

Efraimidis, P. S., & Spirakis, P. G. (2006). Weighted random sampling with a reservoir. *Information Processing Letters*, *97*(5), 181–185. doi: 10.1016/j.ipl.2005.11.003

Efron, B., & Morris, C. (1977). Stein's Paradox in Statistics. *Scientific American*, *236*, 119–127. doi: 10.1038/scientificamerican0577-119

Emmons, K. M., Wechsler, H., Dowdall, G., & Abraham, M. (1998). Predictors of smoking among us college students. *American journal of public health*, *88*(1), 104–7. doi: 10.2105/AJPH.88.1.104

Escobar, L., & Moser, E. (1993). A Note on the Updating of Regression Estimates. *The American Statistician*, *47*(3), 192–194. doi: 10.2307/2684974

Gaber, M. M. (2012). Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 79–85. doi: 10.1002/widm.52

Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams: A review. *SIGMOD*, *34*(2), 18–26. doi: 10.1145/1083784.1083789

Gelman, A. (2007). Rich State, Poor State, Red State, Blue State: What's the Matter with Connecticut? *Quarterly Journal of Political Science*, *2*(4), 345–367. doi: 10.1561/100.00006026

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian Data Analysis* (2nd ed.). Chapman and Hall/CRC.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press. doi: 10.2277/0521867061

Goldstein, H. (1986). Multilevel mixed linear model analysis using iterative generalized least squares. *Biometrika*, *73*(1), 43–56. doi: 10.1093/biomet/73.1.43

Goldstein, H., & McDonald, R. P. (1988). A general model for the analysis of multilevel data. *Psychometrika*, *53*(4), 455–467. doi: 10.1007/BF02294400

Goodman, J., & Blum, T. (1996). Assessing the non-random sampling effects of subject attrition in longitudinal research. *Journal of Management*, *22*(4), 627–652. doi: 10.1016/S0149-2063(96)90027-6

Hamaker, E. L., & Wichers, M. (2017). No time like the present. *Current Directions in Psychological Science*, *26*(1), 10-15. doi: 10.1177/0963721416666518

Hofmann, M., & Klinkenberg, R. (2013). *RapidMiner: Data Mining Use Cases and Business Analytics Applications.* Boca Raton, Florida, USA: Chapman & Hall/CRC.

Hofmann, W., Adriaanse, M., Vohs, K. D., & Baumeister, R. F. (2014). Dieting and the self-control of eating in everyday environments: An experience sampling study. *British Journal of Health Psychology*, *19*(3), 523–539. doi: 10.1111/bjhp.12053

Hsu, D. J., Karampatziakis, N., Langford, J., & Smola, A. J. (2011). Parallel online learning. *CoRR*, *abs/1103.4204*. Retrieved from `http://arxiv.org/abs/1103.4204`

Ippel, L., Kaptein, M. C., & Vermunt, J. K. (2016a). Dealing with Data Streams: an Online, Row-by-Row, Estimation Tutorial. *Methodology*, *12*, 124—138. doi: 10.1027/1614-2241/a000116

Ippel, L., Kaptein, M. C., & Vermunt, J. K. (2016b). Estimating random-intercept models on data streams. *Computational Statistics & Data Analysis*, *104*, 169–182. doi: 10.1016/j.csda.2016.06.008

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. Springer New York. doi: 10.1007/978-1-4614-7138-7

James, W., & Stein, C. (1961). Estimation with Quadratic Loss. In J. Neyman (Ed.),

Proceedings of the fourth berkeley symposium on mathematical statistics and probability, volume 1: Contributions to the theory of statistics (Vol. 1, pp. 361–379). Berkeley, Calif: University of California Press. doi: 10.1007/978-1-4612-0919-5_30

John, L. K., Loewenstein, G., & Prelec, D. (2012). Measuring the Prevalence of Questionable Research Practices With Incentives for Truth Telling. *Psychological Science*, *23*(5), 524–532. doi: 10.1177/0956797611430953

Kabisa (Tchumtchoua), S., Dunson, D. B., & Morris, J. S. (2016). Online variational bayes inference for high-dimensional correlated data. *Journal of Computational and Graphical Statistics*, *25*(2), 426-444. doi: 10.1080/10618600.2014.998336

Kaptein, M. (2014). {RStorm}: Developing and Testing Streaming Algorithms in {R}. *The R Journal*, *6*(1), 123–132.

Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). *Learning Spark: Ligtning-Fast Big Data Analysis*. O'Reilly Media.

Keith, T. (2014). *Multiple regression and beyond* (2nd ed.). Routledge. doi: 10.4324/9781315749099

Kenny, D. A., & Judd, C. M. (1986). Consequences of violating the independence assumption in analysis of variance. *Psychological Bulletin*, *99*(3), 422–431. doi: 10.1037/0033-2909.99.3.422

Killingsworth, M. A., & Gilbert, D. T. (2010). A Wandering Mind Is an Unhappy Mind. *Science*, *330*(6006), 932. doi: 10.1126/science.1192439

Kooreman, P., & Scherpenzeel, A. (2014). High frequency body mass measurement, feedback, and health behaviors. *Economics & Human Biology*, *14*, 141–153. doi: 10.1016/j.ehb.2013.12.003

Lee, J., Podlaseck, M., Schonberg, E., & Hoch, R. (2001). Visualization and Analysis of Clickstream Data of Online Stores for Understanding Web Merchandising. *Data Mining and Knowledge Discovery*, *5*, 59–84. doi: 10.1023/a:1009843912662

Leeuw, E. D. D. (2005). To Mix or Not to Mix Data Collection Modes in Surveys. *Journal of Official Statistics*, *21*(2), 233–255.

Liang, P., & Klein, D. (2009). Online EM for unsupervised models. *Proceedings of Human Language Technologies The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on NAACL 09*, 611. doi: 10.3115/1620754.1620843

Linares, B., Guizar, J. M., Amador, N., Garcia, A., Miranda, V., Perez, J. R., & Chapela, R. (2010). Impact of air pollution on pulmonary function and respiratory symptoms in children. Longitudinal repeated-measures study. *BMC Pulmonary Medicine*, *10*(1), 62. doi: 10.1186/1471-2466-10-62

Liu, Z., Almhana, J., Choulakian, V., & McGorman, R. (2006). Online EM algorithm for mixture with application to internet traffic modeling. *Computational Statistics & Data Analysis*, *50*(4), 1052–1071. doi: 10.1016/j.csda.2004.11.002

Lynch, S. M. (2007). *Introduction to applied bayesian statistics and estimation for social scientists*. Springer New York. doi: 10.1007/978-0-387-71265-9

Manzo, A. N., & Burke, J. M. (2012). Increasing response rate in web-based/internet

surveys. In L. Gideon (Ed.), *Handbook of survey methodology for the social sciences* (pp. 327–343). New York, NY: Springer New York. doi: 10.1007/978-1-4614-3876-2_19

Marz, N., & Warren, J. (2013). *Big Data: Principles and best practices of scalable realtime data systems*. Greenwich, CT, USA: Manning Publications.

McLachlan, G., & Krishnan, T. (2007). Extensions of the EM algorithm. In *Wiley series in probability and statistics* (2nd ed., pp. 159–218). John Wiley & Sons, Inc. doi: 10.1002/9780470191613.ch5

McLachlan, G., & Peel, D. (2000). *Finite Mixture Models*. New York, USA: Wiley series in probability and statistics. doi: 10.1002/0471721182

Moerbeek, M., Van Breukelen, G., & Berger, M. (2003). A comparison of Estimation Methods for Multilevel Logistic Models. *Computational Statistics*, *18*, 19–37. doi: 10.1007/s001800300130

Morris, C., & Lysy, M. (2012). Shrinkage Estimation in Multilevel Normal Models. *Statistical Science*, *27*(1), 115–134. doi: 10.1214/11-STS363

Murnaghan, D. A., Sihvonen, M., Leatherdale, S. T., & Kekki, P. (2007). The relationship between school-based smoking policies and prevention programs on smoking behavior among grade 12 students in Prince Edward Island: A multilevel analysis. *Preventive Medicine*, *44*(4), 317–322. doi: 10.1016/j.ypmed.2007.01.003

Myung, I. (2003). Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, *47*, 90–100. doi: 10.1016/S0022-2496(02)00028-7

Neal, R., & Hinton, G. E. (1998). A View Of The Em Algorithm That Justifies Incremental, Sparse, And Other Variants. In *Learning in graphical models* (pp. 355–368). doi: 10.1007/978-94-011-5014-9_12

Neumeyer, L., Robbins, B., Nair, A., & Kesari, A. (2010). S4: Distributed stream computing platform. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 170–177. doi: 10.1109/ICDMW.2010.172

Ng, S. K., & McLachlan, G. (2003). On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. *Statistics and Computing*, *13*(1), 45–55. doi: 10.1023/A:1021987710829

Opper, M. (1998). A Bayesian Approach to Online Learning. In D. Saad (Ed.), *On-line learning in neural networks* (pp. 363–378). Cambridge: Cambridge University Press. doi: 10.1017/cbo9780511569920.017

Owen, A. B., & Eckles, D. (2012). Bootstrapping data arrays of arbitrary order. *The Annals of Applied Statistics*, *6*(3), 895–927. doi: 10.1214/12-aoas547

Patidar, R., & Sharma, L. (2011). Credit Card Fraud Detection Using Neural Network. *International Journal of Soft Computing and Engineering*, *1*(June), 32–38.

Pebay, P. P. (2008). *Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments.* (Tech. Rep.). doi: 10.2172/1028931

Pébay, P. P., Terriberry, T. B., Kolla, H., & Bennett, J. (2016). Numerically stable,

scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights. *Computational Statistics*. doi: 10.1007/s00180-015-0637-z

Pedro, S., Z., M. O., Baker, R., Bowers, A. J., & Heffernan, N. T. (2013). Predicting college enrollment from student interaction with an intelligent tutoring system in middle school. *Proceedings of the 6th International Conference on Educational Data Mining*, 177–184.

Plackett, R. (1950). Some Theorems in Least Squares. *Biometrika*, *37*(1/2), 149–157. doi: 10.2307/2332158

Powell, M. J. (2009). The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*.

Quintelier, E. (2010). The effect of schools on political participation: a multilevel logistic analysis. *Research Papers in Education*, *25*(2), 137–154. doi: 10.1080/02671520802524810

R Core Team. (2013). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria.

R Core Team. (2016). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria.

Rabe-Hesketh, S., Skrondal, A., & Pickles, A. (2002). Reliable estimation of generalized linear mixed models using adaptive quadrature. *Stata Journal*, *2*(1), 1–21.

Raudenbush, S., & Bryk, A. (2002). *Hierarchical Linear Models: applications and Data Analysis Methods* (2nd ed.; J. de Leeuw, Ed.). Thousand Oaks, California, USA: Sage Pulication.

Robert, C. P. (2015). The Metropolis-Hastings algorithm. *ArXiv e-prints: 1504.01896*.

Rubin, D., & Thayer, D. T. (1982). EM algorithms for ML factor analysis. *Psychometrika*, *47*(1), 69–76. doi: 10.1007/BF02293851

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. *International Conference on Collaboration Technologies and Systems (CTS)*, 42–47. doi: 10.1109/CTS.2013.6567202

Schaul, T., Zhang, S., & LeCun, Y. (2013). No More Pesky Learning Rates. *Journal of Machine Learning Research*, *28*(3), 343–351.

Shalev-Shwartz, S. (2011). Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning*, *4*(2), 107–194. doi: 10.1561/2200000018

Sherman, J., & Morrison, W. J. (1950). Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics*, *21*(1), 124–127. doi: 10.1214/aoms/1177729893

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant. *Psychological Science*, *22*(11), 1359–1366. doi:

10.1177/0956797611417632

Skrondal, A., & Rabe-Hesketh, S. (2004). *Generalized latent variable models: multilevel, longitudinal, and structural equation models* (Vol. 17). Chapman and Hall/CRC. doi: 10.1201/9780203489437

Steenbergen, M., & Jones, B. (2002). Modeling Multilevel Data Structures. *American Journal of Political Science*, *46*(1), 218–237. doi: 10.2307/3088424

Stein, C. (1956). Inadmissibility of the Usual Estimator for the Mean of a Multi-Variate Normal Distribution. In *Proc. third berkeley symp. on math. statist. and prob* (Vol. 1, pp. 197–206).

Steiner, P. M., & Hudec, M. (2007). Classification of large data sets with mixture models via sufficient EM. *Computational Statistics and Data Analysis*, *51*(11), 5416–5428. doi: 10.1016/j.csda.2006.09.014

Strube, M. J. (2006). SNOOP: a program for demonstrating the consequences of premature and repeated null hypothesis testing. *Behavior research methods*, *38*(1), 24–27. doi: 10.3758/BF03192746

Swendsen, J., Ben-Zeev, D., & Granholm, E. (2011). Real-time electronic ambulatory monitoring of substance use and symptom expression in schizophrenia. *American Journal of Psychiatry*, *168*(2), 202–209. doi: 10.1176/appi.ajp.2010.10030463

Tarrès, P., & Yao, Y. (2014). Online Learning as Stochastic Approximation of Regularization Paths: Optimality and Almost-Sure Convergence. *IEEE Transactions on Information Theory*, *60*(9), 5716 – 5735. doi: 10.1109/TIT.2014.2332531

Thiesson, B., Meek, C., & Heckerman, D. (2001). Accelerating EM for large databases. *Machine Learning*, *45*(3), 279–299. doi: 10.1023/A:1017986506241

Toshniwal, A., Donham, J., Bhagat, N., Mittal, S., Ryaboy, D., Taneja, S., … Fu, M. (2014). Storm@twitter. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, 147–156. doi: 10.1145/2588555.2595641

Trull, T. J., & Ebner-Priemer, U. W. (2009). Using Experience Sampling Methods/Ecological Momentary Assessment (ESM/EMA) in Clinical Assessment and Clinical Research: Introduction to the Special Section. *Psychological Assessment*, *21*(4), 457–462. doi: 10.1037/a0017653

Turaga, D., Andrade, H., Gedik, B., Venkatramani, C., Verscheure, O., Harris, J. D., … Jones, P. (2010). Design principles for developing stream processing applications. *Software: Practice and Experience*, *40*(12), 1073–1104. doi: 10.1002/spe.993

van der Palm, D. W., van der Ark, L. A., & Vermunt, J. K. (2016). A comparison of incomplete-data methods for categorical data. *Statistical Methods in Medical Research*, *25*(2), 754–774. doi: 10.1177/0962280212465502

van de Schoot, R., Kaplan, D., Denissen, J., Asendorpf, J. B., Neyer, F. J., & van Aken, M. A. (2014). A gentle introduction to bayesian analysis: Applications to developmental research. *Child Development*, *85*(3), 842–860. doi: 10.1111/cdev.12169

Welford, B. (1962). Note on a Method for Calculating Corrected Sums of Squares

and Products. *Technometrics*, *4*(3), 419–420. doi: 10.2307/1266577

Whalen, C. K., Jamner, L. D., Henker, B., Delfino, R. J., & Lozano, J. M. (2002). The ADHD spectrum and everyday life: experience sampling of adolescent moods, activities, smoking, and drinking. *Child development*, *73*(1), 209–27. doi: 10.1111/1467-8624.00401

Wilkinson, L., & Task Force on Statistical Inference. (1999). Statistical methods in psychology journals: Guidelines and explanations. *American Psychologist*, *54*(8), 594–604. doi: 10.1037/0003-066x.54.8.594

Wilson, D., & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429–1451. doi: 10.1016/S0893-6080(03)00138-2

Witten, I. H., Frank, E., & Hall, M. A. (2013). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann Publishers. doi: 10.1016/c2009-0-19715-5

Wolfe, J., Haghighi, A., & Klein, D. (2008). Fully distributed EM for very large datasets. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 1184–1191. doi: 10.1145/1390156.1390305

Xu, W. (2011). Towards optimal one pass large scale learning with averaged stochastic gradient descent. *CoRR*, *abs/1107.2490*.

Yang, H., Xu, Z., King, I., & Lyu, M. R. (2010). Online learning for group lasso. In *Proceedings of the 27th international conference on machine learning* (pp. 1191–1198). Haifa, Israel.

Yang, Y., & Dunson, D. B. (2013). Sequential Markov Chain Monte Carlo. *ArXiv e-prints*.

Young-Xu, Y., & Chan, K. A. (2008). Pooling overdispersed binomial data to estimate event rate. *BMC medical research methodology*, *8*, 58. doi: 10.1186/1471-2288-8-58

Summary

The technological developments of the last decades, e.g., the introduction of the smartphone, have created opportunities to efficiently collect data of many individuals over an extensive period of time. While these technologies allow for intensive longitudinal measurements, they also come with new challenges: data sets collected using these technologies could become extremely large, and in many applications the data collection is never truly 'finished'. As a result, the data keep streaming in and analyzing data streams using the standard computation of well-known models becomes inefficient as the computation has to be repeated each time a new data point enters to remain up to date. In this thesis, methods to analyze data streams are developed. The introduction of these methods allows researchers to broaden the scope of their research, by using data streams.

In Chapter 2, multiple approaches for analyzing data streams are discussed, though the main focus of this chapter is on online learning. Online learning means that the parameter estimates are estimated while the data enter, without going back to older data points to update the parameter estimates. In this chapter, the standard computations of several common models for *independent* observations are adapted such that these models could be computed online in data streams. These online computations are illustrated with R code, e.g., to compute correlation and linear regression online. For more complex models that do not have simple (closed-form) computations, Stochastic Gradient Decent is introduced. This method approximates the solution (e.g., the Maximum Likelihood solution), a data point at a time. This optimization method is illustrated by fitting a logistic model to a data stream.

Chapter 2 focuses on data streams consisting of independent observations. However, data streams often consist of observing the same individuals repeatedly. Observing the same individual multiple times creates a nesting in the data. Many statistical models, however, assume that the observations are not nested, or independent. In Chapter 3, four online methods for the analysis of nested observations in data streams are developed. These four methods combine the observations of an individual with the data of all the other individuals, to obtain more accurate predictions than when using only the individual's observations. However, fitting a model that accounts for both nested observations and binary outcomes in a data stream can be computationally challenging. The four methods that are presented in this chapter are based on existing shrinkage factors. The prediction accuracy of the offline and online shrinkage factors is compared in a simulation study. While the existing methods differ in their prediction accuracy, the differences in accuracy between the online

and the offline shrinkage factors are small.

A model-based approach to analyze data streams with dependent observations is discussed in Chapter 4. Data sets with nested structures are typically analyzed using multilevel models. However, in the context of data streams, estimating multilevel models can be challenging: the algorithms used to fit multilevel models repeatedly revisit all data points and, in the case that new data enter, have to redo this procedure to remain up to date. Chapter 4 presents a solution for this problem with the Streaming Expectation Maximization Approximation (SEMA) algorithm for fitting random intercept models online. The performance of SEMA is compared to traditional methods of fitting random intercept models in a simulation study and in an empirical example. The prediction accuracy of SEMA is both competitive and orders of magnitude faster than traditional methods.

Chapter 5 provides an extension of the SEMA algorithm to allow online multilevel modeling with fixed and random effects. In a simulation study, models with random intercepts and slopes and fixed effects on both the level of the individual and the level of observations are included. The SEMA algorithm is able to accurately estimate the parameter values. The performance of SEMA is also illustrated using an empirical example, where individuals' weight is predicted in a data stream. In this example, the prediction accuracy of SEMA and the traditional methods are very similar.

Finally, Chapter 6 discusses the contributions, such as estimating multilevel models efficiently in data streams, and limitations, such as the small scale study of convergence, of the work presented in this thesis. Directions for further research are provided, such as how SEMA could be extended to fit other models as well and how related fields could make use of the work presented in this thesis.

Samenvatting

In de afgelopen decennia hebben er veel technologische ontwikkelingen, bijvoorbeeld de opkomst van de smartphone, plaatsgevonden. Deze technologische ontwikkelingen hebben nieuwe mogelijkheden gecreëerd om over lange periodes data van veel mensen tegelijkertijd te verzamelen. Doordat de dataverzameling nu op een grotere schaal kan plaatsvinden, ontstaan er nieuwe uitdagingen: de databestanden, die verzameld worden met deze methodes, kunnen zeer groot worden. Om deze databestanden te kunnen verwerken en analyseren moet er zowel genoeg opslag- als rekencapaciteit zijn. Een bijkomend probleem is dat het niet altijd duidelijk is wanneer de dataverzameling afgelopen is, aangezien nieuwe data continu het computergeheugen binnen blijven *stromen*. De standaardmethodes om data te analyseren zijn veelal niet geschikt om deze *datastromen* te analyseren: deze methodes veronderstellen namelijk dat alle data tegelijkertijd beschikbaar zijn in het computergeheugen. Hierdoor moet een analyse steeds opnieuw uitgevoerd worden om gebruik te kunnen maken van de nieuw binnengekomen data. In deze thesis worden methodes besproken en ontwikkeld, die 1) het resultaat van een analyse eenvoudig aanpassen aan de hand van nieuwe data, zonder de analyse opnieuw uit te voeren; en 2) het overbodig maken om observaties op te slaan.

Eerst worden de bestaande methoden besproken en geïllustreerd in hoofdstuk 2. De focus van dit hoofdstuk ligt op de *online learning* methode. Deze methode past eenvoudig de uitkomst van een analyse aan wanneer nieuwe data binnen komen, zonder gebruik te maken van de oude data. De data punten hoeven daarom niet opgeslagen te worden. Voor een aantal veelgebruikte modellen wordt (met R code) de aangepaste (online) manier van schatten gepresenteerd (bijvoorbeeld een gemiddelde of een lineaire regressie). De berekening van sommige modellen kan echter niet eenvoudig aangepast worden zodat ze online berekend kunnen worden. Een voorbeeld van zo'n model is het logistische model, dat gebruikt wordt om een binaire uitkomst te voorspellen. Om dit soort modellen toch te kunnen gebruiken in een datastroom zijn meer complexe technieken nodig. Met R code laten we zien hoe Stochastic Gradient Descent, een online benaderingsmethode, een logistisch model schat.

In de sociale wetenschappen bestaan datastromen vaak uit herhaalde metingen van dezelfde personen, bijvoorbeeld of iemand wel of niet op een website op advertenties klikt. Het herhaaldelijk observeren van dezelfde persoon creëert een samenhang (of afhankelijkheid) tussen de observaties die van dezelfde persoon afkomstig zijn: twee observaties van dezelfde persoon lijken waarschijnlijk meer op elkaar dan

twee willekeurige observaties die niet van dezelfde persoon afkomstig zijn. Echter, voor observaties die *afhankelijk* zijn, is het schatten van de daarvoor geschikte modellen complex, met name als de uitkomst binair is. In hoofdstuk 3 worden vier online methodes ontwikkeld, die rekening houden met de samenhang tussen de observaties van dezelfde persoon en met een binaire uitkomst. De nieuwe online methodes zijn gebaseerd op reeds bestaande *shrinkage* methodes. Shrinkage methodes combineren de observaties van een persoon met de data van alle andere personen. Op deze manier worden er accuratere voorspellingen gemaakt dan voorspellingen die alleen gebaseerd zijn op de observaties van iedere persoon afzonderlijk. Uit een simulatiestudie blijkt dat er nauwelijks verschillen zijn tussen de online manier versus de standaard manier van schatten van de shrinkage methodes.

In het volgende hoofdstuk wordt een methode besproken die gebaseerd is op een veelgebruikt model wanneer de observaties gegroepeerd zijn. Het model heet het multilevel model: het lagere level (level 1) duidt de observaties aan en het hogere level (level 2) geeft de personen weer. Het schatten van zo'n multilevel model is lastig in een datastroom, omdat de gebruikelijke methodes alle data in het computer geheugen nodig hebben en herhaaldelijk deze data gebruiken om het model te kunnen schatten. In hoofdstuk 4 wordt een nieuw algoritme ontwikkeld dat het multilevel model kan schatten zonder gebruik te maken van oude observaties. Het algoritme heet SEMA: Streaming Expectation Maximization Approximation en het is gebaseerd op een bestaand algoritme om het multilevel model te schatten. In een simulatie studie en met bestaande data wordt het random intercept model, een eenvoudig multilevel model, geschat. De standaardmethode en het SEMA algoritme presteren vergelijkbaar, terwijl SEMA vele malen sneller is dan de standaardmethode.

Hoofdstuk 5 is een uitbreiding van het SEMA algoritme zodat niet alleen het random intercept model geschat kan worden, maar ook complexere multilevel modellen. Met deze uitbreiding kan SEMA modellen met meerdere fixed effecten en random effecten schatten in een data stroom. Fixed effecten zijn effecten die voor alle personen even groot zijn. Random effecten daarentegen kunnen verschillen per persoon. We laten aan de hand van een simulatiestudie en de analyse van bestaande data zien dat SEMA multilevel modellen met fixed effecten op beide levels, random intercepts en random slopes kan schatten. In beide gevallen is SEMA goed in staat om accurate voorspellingen te maken terwijl de data binnen komt.

Tot slot, in het laatste hoofdstuk worden de bijdrage, zoals multilevel modellen efficiënt kunnen schatten in een data stroom, en limitaties, zoals een beperkte analyse van convergentie, van deze thesis besproken. Verder worden ook richtingen voor vervolgonderzoek besproken zoals bijvoorbeeld hoe het SEMA algoritme nog verder uitgebreid kan worden en hoe gerelateerde velden gebruik kunnen maken van het werk besproken in deze thesis.

Dankwoord

In deze laatste paar pagina's van mijn boekje wil ik graag iedereen bedanken die bijgedragen heeft aan dit proefschrift.

Vier jaar geleden heb ik van Jeroen Vermunt de kans gekregen om aan dit project te beginnen. Hij stelde voor om eens met Maurits Kaptein te praten en Jeroen bood me een promotieplek binnen het Methoden en Technieken departement. Jeroen, ik ben je dankbaar dat je me deze kans gegeven hebt. Ik heb je feedback altijd erg gewaardeerd en daar waar ik er zelf niet uitkwam of de zaken toch net niet helemaal scherp had, kon ik altijd rekenen op je uitleg.

Maurits, eigenlijk ben je de afgelopen vijf jaar mijn begeleider geweest, want ik mocht ook mijn master thesis bij jou schrijven. In deze tijd heb ik ontzettend veel van je mogen leren en je stond altijd voor me klaar. Daar waar mogelijk, probeerde je bij al mijn presentaties te zijn. Zelfs even twee uur naar Kerkrade rijden, was blijkbaar geen probleem. Hoewel ik zo vlak voor een presentatie het wellicht niet altijd heb laten blijken, heb ik het enorm gewaardeerd dat je erbij zat. En in de tijden dat het me allemaal even niet mee zat, wist ik ook op persoonlijk vlak dat je er voor me was. Je hebt het veilig voor me gemaakt om fouten te maken en iedereen die mij ook maar een beetje kent, weet dat dat een van de grootste complimenten is die ik je kan geven.

I also want to thank my two hosts at Berkeley, Sophia Rabe-Hesketh and Anders Skrondal. I am grateful to have gotten the opportunity to learn from you and from the QME students you supervise. A special thanks to James Mason and Joonho Lee with whom I shared many dinners and ideas!

Buiten mijn twee vaste begeleiders, kon ik altijd rekenen op de hulp en uitleg van de andere collega's binnen het departement. Marcel van Assen, bedankt voor de 'spar'-momentjes wanneer we het eigenlijk alle twee niet echt wisten maar je me toch weer op weg wist te helpen. Marcel Croon, bedankt dat u plaats wilt nemen in mijn commissie. Uw deur stond altijd voor me open en uw wiskundige kennis heeft me vele malen vooruit geholpen. Katrijn, ik kon altijd bij je aankloppen of ik nu een presentatie wilde oefenen of ergens vastliep in mijn papers, bedankt!

Verder wil ik graag mijn collega's bedanken voor alle gezelligheid op kantoor, de borrels, tijdens de IOPS cursussen en conferenties. Mijn kantoorgenootjes Robbie, voor de vele doorgestuurde linkjes naar alle financiele documenten die ik nooit kon vinden, en de R support waar ik altijd kon rekenen, Chris en Niek voor alle discussies en vele koppen koffie, mede namens mijn koffieverslaving, bedankt! Mede-VICI

leden, zowel de huidige als de voormalige: Margot, Zsuzsa, Daniel O., Jeroen, Katrijn, Kim, Laura, Davide, Mattis, Erwin, Niek, Geert, Leonie, Pia, Reza, ik heb veel van jullie papers en jullie feedback mogen leren. Uiteraard vergeet ik de Bayes club met Jeroen, Jesper, Joris, Florian, Davide, Geert, Dino, en Sara ook niet!

Onderwijs geven is een van die taken waar ik elk jaar weer naar uitkeek, tot ongeveer halverwege het blok dan "was ik er wel weer klaar mee" en wilde ik "weer gewoon m'n werk doen". Leoni, Josine, Jules, Reza, Eva, Hannah, Chris, Inga, Katrijn, John en Luc, ik heb erg genoten om met jullie het onderwijs te verzorgen. Wilco, bedankt voor de organisatie van het onderwijs en dat ik bij je terecht kon met alle vragen, opmerkingen en frustraties. Guy, als groentje ben ik bij jouw vak MTO-A-MAW begonnen 7 jaar geleden, bedankt voor het vertrouwen en de support ook in de vele jaren daarna.

Verder wil ik nog de secretaresses van MTO bedanken, Marieke, Liesbeth, en Anne-Marie. Bedankt voor alle support en al het geregel! Zonder jullie was ik nu waarschijnlijk nog steeds aan het zoeken naar de juiste formulieren.

Naast mijn collega's wil ik ook mijn vrienden bedanken. Allereerst mijn paranymfen Erwin en Hilde, bedankt dat jullie straks achter mij willen gaan staan. Bedankt dat jullie steeds opnieuw voor me klaar staan. Ook al is het straks wellicht wat meer uit het oog, zeker niet uit het hart.

Tom, we zijn het nooit met elkaar eens, toch heb ik veel van je geleerd, en je schrijven is iets waar ik altijd jaloers op zal zijn.

Daniel Pineda, Drew, Matt, and Colin a big thank you for welcoming me into your house. Thanks for giving me the full American experience by inviting me to the Super Bowl party and the 'lovely' election season. Daniel, thank you for showing me around.

Continuing in English, Adam thank you for sharing your stories and always offering a listening ear. Greta, I am already looking forward to our next city trip.

Aan mijn tijd bij Rataplan heb ik veel vrienden overgehouden. Janneke, Niek, Liselot, Letje, Peppie, Sanne, Fons, Hilde en Marius. Wat heb ik met jullie geweldige tijden beleefd, op het terras, in de langeboom, op de vloer, kanoën, NSK, etc. etc. etc.! Door jullie weet ik dat er meer bestaat dan alleen mijn proefschrift. Sanne, met je mooie gezinnetje, dankjewel voor je vriendschap, je bent me dierbaar. Hilde en Marius, bedankt voor de klustherapie om alle phd en gerelateerde stress van me af te zetten, jullie zijn fantastisch!

Ik wil mijn familie bedanken, pa, ma, Marco, Agnita en Lucas. Bedankt voor jullie steun in de afgelopen jaren.

Jeroen, bedankt voor je geduld en samen met jou in Mestreech, kump alles good!